AN ALTERNATING DIRECTION GALERKIN METHOD FOR NONLINEAR PARABOLIC PROBLEMS

John Mark Franklin

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN ALTERNATING DIRECTION GALERKIN METHOD
FOR NONLINEAR PARABOLIC PROBLEMS

by

John Mark Franklin

June 1975

Thesis Advisor: D. Archer

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Alternating Direction Galerkin Method for Nonlinear Parabolic Problems		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John Mark Franklin		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1975 13. NUMBER OF PAGES 76
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An implementation of the Laplace modified centered difference Galerkin method for the solution of the general non-linear parabolic differential equation. Alternating directions methods are used to approximate the solution in

(20. ABSTRACT Continued)

two dimensions for a rectangle. Hermite cubics are used as basis functions for the space.

An Alternating Direction Galerkin Method for Nonlinear Parabolic Problems

by

John Mark Franklin

Ensign, United States Navy

B.S., United States Naval Academy, 1974

Submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE WITH MAJOR IN MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1975

ABSTRACT

An implementation of the Laplace modified centered difference Galerkin method for the solution of the general non-linear parabolic differential equation. Alternating directions methods are used to approximate the solution in two dimensions for a rectangle. Hermite cubics are used as basis functions for the space.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Lieutenant David Archer for his patience and aid in the preparation of this thesis. The author would also like to thank Dr. Richard Franke for his help in the preparation of the text, Dr. T. Cooper for help in presenting the test problems.

The author also wishes to thank his mother, whose encouragement throughout his life has helped him reach this point.

I. INTRODUCTION

The parabolic partial differential equation

$$\frac{\partial u}{\partial t} = \nabla\cdot(a(x,t,u)\nabla u) + f(x,t,u,\nabla u) \qquad (x,t) \in \Omega\times(0,t]$$

has been the subject of the development of several numerical approximation techniques. The equation is known as the heat equation in a specialized form and is useful in the study of heat transfer and thermodynamics in general. It also has applications in other fields of engineering and science, such as fluid dynamics and meteorology.

The development of Galerkin methods for the solution of a general parabolic and hyperbolic equation has been relatively recent [3,6,9]. This thesis describes a program for the solution of the general parabolic problem using the central different Laplace modified alternating direction Galerkin scheme described in [5,6,8]. The program was built in steps beginning with the one dimensional case and then extended to the two dimensional case.

The program solves the problem for a rectangular domain Ω, and uses nine Gauss quadrature points for integration in the plane. It also allows for nonlinearities in the forcing function $f(x,t,u,\nabla u)$ and the function $a(x,t,u)$ can be extended to have nonlinear terms of ∇u. The theoretical proofs of stability and convergence are available for these cases in [4].

The program was written in the Fortran IV computer language and tested on an IBM 360/67 computer system.

II. DEFINITIONS, NOTATION, AND PRELIMINARY CONCEPTS

In the following section several concepts and definitions which are essential for the understanding of the thesis are presented. Two specific spaces of functions will be considered and definitions of the norms in the spaces will be given.

Let $\Omega \subset R^n$, with smooth boundary $\partial\Omega$, be the domain of interest for the function spaces. The first of these spaces is that of square integrable or measurable functions, $L^2(\Omega)$. If $w \in L^2(\Omega)$, then the norm of w is defined as

$$\|w\|_{L^2(\Omega)} = \int_\Omega w^2 \, dx < \infty \quad .$$

$L^2(\Omega)$ is a Hilbert space with respect to the inner product

$$\langle f,g \rangle = \int_\Omega f(x) \, g(x) \, dx$$

The Sobolev space $H^0(\Omega)$ is equivalent to $L^2(\Omega)$.

The second space considered will be the Sobolev space $H^1(\Omega)$, defined as

$$H^1 = \{w | D^p w \in L^2(\Omega), \ |p| = 0,1\}$$

where

$$D^p = \frac{\partial^p}{\partial x_1^{p_1} \ \cdots \ \partial x_n^{p_n}} \quad , \quad |p| = \sum_{i=1}^{n} (p_i)$$

The norm of w is defined as

$$\|w\|_{H^1(\Omega)} = [\, \|w\|^2_{L^2(\Omega)} + \sum_{i=1}^{n} \|\frac{\partial w}{\partial x_i}\|^2_{L^2(\Omega)} \,]^{1/2} .$$

Of special interest for the development of this paper is the space H_0^1 a subspace of H^1 defined as

$$H_0^1 = \{w \,|\, w \in H^1(\Omega) \quad \text{and} \quad w = 0, \text{ for all } x \in \partial\Omega\}$$

The norm for this subspace is defined to be

$$\|w\|_{H_0^1(\Omega)} = [\, \sum_{i=1}^{n} \|\frac{\partial w}{\partial x_i}\|^2_{L^2(\Omega)} \,]^{1/2}$$

A major concept which is used in the formulation of Galerkin's method is that of the weak solution to a differential equation. Let L be the differential operator describing the differential equation, that is

$$Lu = f \tag{2.1}$$

A function u is said to satisfy the weak form of equation (2.1), if for every test function v in some test space

$$\langle Lu, v \rangle = \langle f, v \rangle \tag{2.2}$$

If u is not sufficiently differentiable to be substituted in (2.1), but does satisfy (2.2), then u is said to be a weak solution.

To illustrate the weak solution, consider

$$Lu = u_t - u_{xx} = f(x,t) \quad , \quad x \in I = (0,1), \quad t > 0.$$

$$u(0,t) = u(1,t) = 0 \quad ; \quad u(x,0) = u_0(x) \quad , \quad x \in I \tag{2.3}$$

The weak form of this equation is

$$\langle u_t - u_{xx}, v \rangle = \langle f, v \rangle \quad \text{for all } v \in H_0^1(I) \tag{2.4}$$

$$u(0,t) = u(1,t) = 0$$

$$u(x,0) = u_0(x)$$

Integrating by parts results in

$$\langle u_t, v \rangle + \langle u_x, v_x \rangle = \langle f, v \rangle \quad \text{for all } v \in H_0^1(I)$$

$$u(x,0) = u_0(x) \tag{2.5}$$

Note that $u \in H_0^1(I)$ can be a solution to (2.5), that is a weak solution of (2.3), however a strong solution of (2.3) must satisfy $u \in C^2(I)$ for each t.

III. THE FORMULATION OF THE GALERKIN METHOD

The formulation of Galerkin methods for the solution of differential equations is based on the weak form of the equation. Consider the nonlinear parabolic equation

$$\begin{aligned} &\frac{\partial u}{\partial t} = \nabla\cdot(a(x,t,u)\nabla u) + f(x,t,u,\nabla u) \quad , \ (x,t) \in \Omega\times(0,T] \\ &u(x,0) = u_0(x) \quad , \quad x \in \Omega \\ &u(x,t) = 0 \qquad (x,t) \in \partial\Omega\times(0,T] \end{aligned} \tag{3.1}$$

and the test space $H_0^1(\Omega)$. Clearly for any $v \in H_0^1(\Omega)$

$$\frac{\partial u}{\partial t} v = \nabla\cdot(a(x,t,u)\nabla u)v + f(x,t,u,\nabla u)v$$

Taking the integral over the region Ω yields

$$\int_\Omega \frac{\partial u}{\partial t} v \, dx = \int_\Omega \nabla\cdot(a(x,t,u)\nabla u)v \, dx + \int_\Omega f(x,t,u,\nabla u)v \, dx$$

Using Green's first formula for integration by parts in multiple dimensions results in

$$\int_\Omega \frac{\partial u}{\partial t} v \, dx = \int_\Omega a(x,t,u)\nabla u \, v \, dx - \int_\Omega a(x,t,u)\nabla u\cdot\nabla v \, dx + \int_\Omega f(x,t,u,\nabla u)v \, dx \ .$$

Since $v = 0$ on $\partial\Omega$ it follows that

$$\int_{\partial\Omega} a(x,t,u)\nabla u\, v\, dx = 0$$

and

$$\int_{\Omega} \frac{\partial u}{\partial t}\, v\, dx + \int_{\Omega} a(x,t,u)\nabla u\cdot\nabla v\, dx = \int_{\Omega} f(x,t,u,\nabla u)\, v\, dx \quad .$$

Therefore u satisfies

$$<\frac{\partial u}{\partial t}, v> + <a(x,t,u)\nabla u, \nabla v> = <f(x,t,u,\nabla u), v> \qquad (3.2)$$

$$<u(x,0), v> = <u_0, v> \qquad \text{for all } v \in H_0^1(\Omega) \ .$$

Let M be a finite dimensional subspace of $H_0^1(\Omega)$. The continuous-time Galerkin method is to find for each $t \in (0,T]$ a differentiable map $U(\cdot,t):[0,T] \to M$, such that

$$<\frac{\partial U}{\partial t}, V> + <a(x,t,U)\nabla U, \nabla V> = <f(x,t,U,\nabla U), V> \qquad (3.3)$$

$$<U,V> = <u_0, V> \ , \quad t = 0 \ , \quad V \in M$$

Now let M be a subspace of $H_0^1(\Omega)$ such that $M = \text{Span}(w_1, w_2, \ldots, w_n)$, where $\{w_i\}_{i=1}^n$ is a linearly independent set. Under these conditions it is possible to write

$$U(x,t) = \sum_{j=1}^{n} \alpha_j(t)\, w_j(x) \quad .$$

Substituting in equation (3.3) and letting $V = w_i$

$$< \sum_{j=1}^{n} \alpha_j(t) w_j(x), w_i(x) > + < a(x,t,U) \sum_{j=1}^{n} \alpha_j(t) \nabla w_j(x), \nabla w_i(x) >$$

$$= < f(x,t,U,\nabla U), w_i(x) >$$

and

$$< \sum_{j=1}^{n} \alpha_j(0) w_j(x) > = < u_0, w_i(x) >$$

Since integration is a linear operation and the inner product defined is not over the time space, it follows that,

$$\sum_{j=1}^{n} \alpha_j'(t) < w_j(x), w_i(x) > + \sum_{j=1}^{n} \alpha_j(t) < a(x,t,U) \nabla w_j(x), \nabla w_i(x) >$$

$$= < f(x,t,U,\nabla U), w_i(x) > \qquad (3.4a)$$

and

$$\sum_{j=1}^{n} \alpha_j(0) < w_j(x), w_i(x) > = < u_0, w_i(x) > \quad i=1,2,\ldots,n \qquad (3.4b)$$

Equations (3.4a) and (3.4b) can be written as a system of n-nonlinear ordinary differential equations

$$M\alpha'(t) + S(\alpha)\alpha(t) = F$$

$$M\alpha(0) = b \qquad (3.5)$$

where $S(\alpha) = (s_{ij})$ and $M = (m_{ij})$ are n^{th} order matrices and

$$s_{ij} = \langle a(x,t,U)\nabla w_j(x), \nabla w_i(x)\rangle$$

$$m_{ij} = \langle w_j(x), w_i(x)\rangle$$

and

$$F = (f_i) \; ; \quad f_i = \langle f(x,t,U,\nabla U), w_i(x)\rangle$$

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$$

$$b = (\langle u_0, w_1\rangle, \langle u_0, w_2\rangle, \ldots, \langle u_0, w_n\rangle)^T$$

Using the properties of the functions w_i, it can be shown that the matrix M is positive definite, and with the additional property that a(x,t,u) is bounded above and below, the matrix S can also be shown to be positive definite. If in addition a(x,t,u) and f(x,t,u,∇u) satisfy Lipschitz conditions, it can be shown that the system of differential equations has a unique solution.

In the one dimensional case, an approximate solution to equation (3.5a) can be obtained by combining it with a finite difference scheme to discretize in time as follows. Assuming, for simplicity, that a(x,t,u) = a = constant and $f(x,t,u,u_x) = f(x,t)$

$$M\left(\frac{\alpha^{m+1} - \alpha^m}{\Delta t}\right) + S\left(\frac{\alpha^{m+1} + \alpha^m}{2}\right) = F^{m+\frac{1}{2}} \qquad (3.6)$$

This is a system of n equations and n unknowns and can be simplified to

$$(2M+\Delta tS)\alpha^{m+1} = (2M-\Delta tS)\alpha^{m} + 2\Delta tF^{m+\frac{1}{2}}$$

or

$$(2M+\Delta tS)(\alpha^{m+1} - \alpha^{m}) = -2\Delta tS\alpha^{m} + 2\Delta tF^{m+\frac{1}{2}}$$

where $\alpha_j^m \simeq \alpha_j(t_m)$ and $F_i^{m+\frac{1}{2}} = \langle f(t_{m+\frac{1}{2}}), w_i \rangle$

In the above equation the time variable t has been discretized to $t_m = m\Delta t$, where $t = T/N$, N a positive integer. This method is known as the Crank-Nicholson-Galerkin method.

The one dimensional program was written to set up the matrices defined above and solve the algebraic equation (3.7). A minor alteration in the program would allow the use of (3.7a), and give the advantage of reducing the computational roundoff error. The program is also limited to the linearity simplifications made above.

The required accuracy of the method in the two dimensional case leads to a large number of basis functions for the space and consequently would require excessively large matrices. The matrices have a band structure and the band width of the matrices is dependent on the number of intervals used in the grid set up for the region. These large matrices can be avoided by implementing an alternating direction method, and using a tensor product basis for the space. This tensor

product basis reduces the problem to working with matrices which have a constant band width of seven, in the case of M being Hermite cubic splines. It also greatly simplifies the problem when the region $\Omega \subset R^2$ is rectangular.

Let $M_x = \text{Span}(\xi_1(x), \xi_2(x), \ldots, \xi_{n_x}(x))$ be a basis for one dimension of the space M and $M_y = \text{Span}(\eta_1(y), \eta_2(y), \ldots \eta_{n_y}(y))$ be a basis for the other dimensions of M. M_x and M_y are bases for one dimensional subspaces of $H_0^1(I_x)$ and $H_0^1(I_y)$, such that $M = M_x \otimes M_y = \text{Span}(\xi_1\eta_1, \xi_1\eta_2, \ldots, \xi_1\eta_{n_y}, \ldots, \xi_{n_x}\eta_{n_y})$. Also define

$$\langle f,g \rangle_x = \int_{I_x} fg \, dx$$

$$\langle f,g \rangle_y = \int_{I_y} fg \, dy$$

Since $M_x \otimes M_y$ forms a basis for the space M, we can write

$$U(x,y,t) = \sum_{p=1}^{n_x} \sum_{q=1}^{n_y} \alpha_{pq}(t)(\xi_p(x) \otimes \eta_q(y)) \ .$$

There are several time discrete methods with which to proceed at this point. The program has been designed to use the alternating direction version of the centered difference Laplace modified Galerkin method described in [5,6,8] and defined as

$$< \frac{U^{m+1} - U^{m-1}}{2\Delta t} , V> + <a^m(x,t,U)\nabla U^m, \nabla V>$$

$$+ 2\lambda^2 \Delta t < \frac{\partial^2}{\partial x \partial y} \partial^2 U^m, \frac{\partial^2 V}{\partial x \partial y}> = <f^m(x,t,U,\nabla U), V>$$

and

$$<U,V> = <u_0,V> \qquad V \in M_x \otimes M_y \tag{3.7}$$

where the superscript of the functions denotes the time step at which it is to be evaluated, and

$$\partial^2 U^m = U^{m+1} - 2U^m + U^{m-1}$$

and

$$\frac{\partial^2 V}{\partial x \partial y} \in L^2(\Omega) \quad .$$

λ is subject to the restriction

$$\lambda > \frac{1}{4} a_{max} = \frac{1}{4} \max_{\substack{x \ \Omega \\ t>0}} a(x,t,u(x,t))$$

for stability [5,6].

Equation (3.7) can be written in the algebraic tensor product form

$$(C_x + 2\lambda\Delta t A_x) \otimes (C_y + 2\lambda\Delta t A_y)\alpha^{m+1} = 2(2\lambda\Delta t(C_x \otimes A_y + A_y \otimes C_y)$$

$$+ 4\lambda^2(\Delta t)^2(A_x \otimes A_y))\alpha^m$$

$$+ (C_x \otimes C_y - 2\lambda\Delta t(C_x \otimes A_y + A_x \otimes C_y) - 4\lambda^2(\Delta t)^2(A_x \otimes A_y))\alpha^{m-1}$$

$$+ 2\Delta t \phi^m$$

where

$$\Phi^m_{pq} = -\langle a^m(x,t,U)\nabla U^m, \nabla(\xi_p \quad \eta_q)\rangle + \langle f^m(x,t,U,\nabla U), \xi_p \quad \eta_q\rangle$$

and

$$C_x = \langle \xi_i, \xi_j \rangle_x \qquad C_y = \langle \eta_i, \eta_j \rangle_y$$

$$A_x = \langle \xi^1_i, \xi^1_j \rangle_x \qquad A_y = \langle \eta^1_i, \eta^1_j \rangle_y$$

Define $e^m = \alpha^m - \alpha^{m-1}$, then equation (3.7) can be written as

$$(C_x+2\lambda\Delta t A_x) \otimes (C_y+2\lambda\Delta t A_y)(e^{m+1}-e^m) = -2(C_x \otimes C_y)e^m + 2\Delta t\Phi^m \tag{3.8a}$$

and

$$\alpha^{m+1} = (e^{m+1}-e^m) + e^m + \alpha^m \tag{3.8b}$$

The scheme (3.8) adds significant efficiency to the program and also reduces roundoff error.

The Laplace centered difference method initially requires two sets of the coefficients, those at time zero and at one time step later. At time zero the set of coefficients are obtained by an $L^2(\Omega)$ projection of the initial conditions into the space, but the second set presents a problem where a variety of methods are available. Two of these methods

were attempted, expanding the initial conditions in a Taylor's Series about t = 0 and using the Laplace forward difference method. The former involved taking the derivatives of the initial condition function and presented a problem when these derivatives resulted in zero. The method which was finally used was the Laplace forward difference scheme. It is defined as follows

$$(C_x + \lambda\Delta t A_x) \otimes (C_y + \lambda\Delta t A_y)(\alpha^{m+1} - \alpha^m) = (\Delta t)\ \Phi^m \qquad (3.9)$$

At first glance the matrices in (3.8a) and (3.9) seem to be different, however in the case of this method λ is under the restriction

$$\lambda > \frac{1}{2} a_{max} = \frac{1}{2} \max a(x,t,u) \qquad x \in \Omega\ ,\quad t > 0$$

thus a prudent choice of λ results in identical matrices and it is not necessary to form a second set of coefficient matrices in the program. It is also noteworthy that Φ^m has the same form as in the previous method and the subroutine used to compute Φ^m may be used for both with a correction of $\frac{1}{2}$ for the forward difference scheme. These advantages led to the final choice.

The tensor product of matrices has the property that

$$(B \otimes D) = (B \otimes I)(I \otimes D)$$

This property enables a significant simplification of two aspects of the programming problem. First, the tensor product multiplication on the right hand side of the equation can be accomplished by computing first

$$(I \otimes C_y) e^m = \gamma^m$$

and then

$$(C_x \otimes I) \gamma^m$$

By properly ordering the vector e^m, this results in a series of multiplication by the relatively small matrices C_x and C_y, that is, matrices which have a constant band width of seven for Hermite cubics rather than a band width which depends on the number of intervals in the grid of the problem as in $C_x \otimes C_y$.

This same idea can be used in solving the linear system. Defining the right hand side of the equations (3.8,3.9) to be β^m, let

$$(I \otimes (C_y + 2\lambda\Delta t A_y))(e^{m+1} - e^m) = \gamma^m$$

Using this, solve first

$$((C_x + 2\lambda\Delta t A_x) \otimes I)\gamma^m = \beta^m$$

and then solve

$$(I \otimes (C_y + 2\lambda\Delta t A_y))(e^{m+1} - e^m) = \gamma^m$$

Again proper ordering of the vectors involved results in the simplification of solving a system of one dimensional problems rather than the larger problem of solving the full tensor product matrix.

The two dimensional program has been designed to solve the problem in the manner described above. The basis functions used in the program are Hermite bicubic splines, taken as the tensor products of one dimensional Hermite cubic splines, and will be described in Section IV. The procedure described above has the drawback that it is in general good only on rectangular polygons, and must be altered significantly to handle regions other than simple rectangles. It can be changed to handle regions which are unions of rectangles [5].

IV. HERMITE BICUBIC SPLINES

The Galerkin procedure is dependent on a set of linearly independent basis functions for the space considered in the problem. Hermite bicubic splines provide such a basis. These splines are a high powered tool for the approximation of smooth functions.

The Hermite bicubic splines are taken as the tensor product of one dimensional Hermite cubics. These cubics have a piecewise polynomial nature, where the polynomial is of degree less than or equal to three. Hermite cubics have globally continuous first derivatives. Thus, the tensor product of the one dimensional cubics has the property that both first partials and the first mixed partial derivatives are continuous. These properties are sufficient for the approximations desired as a result of the program [1].

The package of subroutines described in [2] was used for the program. It consists of subroutines which compute the values of the B-splines at specific points, and given a set of coefficients, computes the value of the approximated function at specified points. The high degree of flexibility of the package was the major reason for its use in the program. The program can be extended to use higher degree B-splines which gives a greater degree of smoothness and accuracy to the approximating function. Also, the extension of the program to higher dimensions can be accomplished by

taking a higher degree tensor product, and making the appropriate alterations in the computational procedures.

Consider the interval [0,1], and the partition of the interval Δ: $0 = x_0 \leq x_1 \leq \ldots \leq x_n = 1$. Let $h_i = x_i - x_{i-1}$, and $I_i = [x_{i-1}, x_i]$. Define $h = \max_i h_i$, and $\underline{h} = \min_i h_i$. The Hermite cubics are piecewise cubics over I_i for all i, and are members of $C^1(I)$. The order of accuracy for Hermite cubics is given by the following well-known theorem:

Theorem. If $u \in H^4(I)$ and $h/\underline{h} \leq \sigma < \infty$, then

$$\inf_{\tilde{u} \in M} \|u - \tilde{u}\|_{H^s(I)} \leq Ch^{4-s} \|u\|_{H^4(I)} \quad , \quad 0 \leq s \leq 3$$

where M is the space of Hermite cubics.

V. ERROR ESTIMATES

The usefulness of a procedure to obtain an approximate solution to a differential equation is measured by its degree of accuracy and the efficiency and costs of its implementation. The purpose of this section is to derive an error bound for the continuous time Galerkin method for a linear version of equation (3.1). The following theorem and its proof is found in [8]. It is restricted to the case where $a(x,t)$ is not a function of u. Proofs for the nonlinear case are also available [3,6,9].

Theorem 1. Let u be the solution to

$$\left\langle \frac{\partial u}{\partial t}, v \right\rangle + \langle a(x,t)\nabla u, \nabla v\rangle = 0 \qquad v \in H_0^1(\Omega), \quad t \in (0,T] \tag{5.1}$$

$$\langle u(x,0)\rangle = \langle u_0(x)\rangle \qquad x \in \Omega$$

and let U be the solution to

$$\left\langle \frac{\partial U}{\partial t}, V \right\rangle + \langle a(x,t)\nabla U, \nabla V\rangle = 0 \qquad V \in M, \quad t \in (0,T] \tag{5.2}$$

$$\langle U, V\rangle = \langle u_0, V\rangle \qquad t = 0$$

and

$$0 < c_0 \leq a_{max}(x,t) \leq c_1 \tag{5.3}$$

Then there exists constants C and ∂, which depend on c_0, c_1, and T, such that

$$\max_{0\le t\le T} \|u-U\|^2_{L^2(\Omega)}(t) + \partial \int_0^\tau \|u-U\|^2_{H_0^1}(t)\, dt$$

$$\le C\Big(\max_{0\le t\le T} \|u-\tilde{u}\|^2_{L^2(\Omega)}(t) + \int_0^\tau \|u-\tilde{u}\|^2_{H_0^1(\Omega)}(t)\, dt$$

$$+ \int_0^\tau \Big\|\frac{\partial}{\partial t}(u-\tilde{u})\Big\|^2_{L^2(\Omega)}(t)\, dt\Big) \qquad (5.4)$$

where $\tilde{u}$ is an arbitrary map of (0,T] into M.

Proof: Replace v with V in (5.1) and subtract (5.2) from (5.1)

$$\Big\langle \frac{\partial u}{\partial t}, V\Big\rangle + \langle a(x,t)\nabla u, \nabla V\rangle = 0$$

$$-\Big\langle \frac{\partial U}{\partial t}, V\Big\rangle + \langle a(x,t)\nabla U, \nabla V\rangle = 0$$

results in

$$\Big\langle \frac{\partial (u-U)}{\partial t}, V\Big\rangle + \langle a(x,t)\nabla(u-U), \nabla V\rangle = 0 \quad .$$

Let $e = u-U$ and let $V = e+\tilde{e}$, where $e = u-\tilde{u}$, then

$$\Big\langle \frac{\partial e}{\partial t}, e+\tilde{e}\Big\rangle + \langle a(x,t)\nabla e, \nabla e+\tilde{e}\rangle = 0$$

Note

$$< \frac{\partial e}{\partial t} , e> = \int_{\Omega} \frac{\partial e}{\partial t} e \, dx$$

$$= \frac{1}{2} \frac{d}{dt} \int_{\Omega} e^2 \, dx$$

$$= \frac{1}{2} \frac{d}{dt} \| e \|^2_{L^2(\Omega)}$$

and

$$<a\nabla e, \nabla e> \geq c_0 \| e \|^2_{H_0^1(\Omega)} \qquad \text{from (5.3)}$$

and

$$|<a\nabla e, \nabla \tilde{e}>| \leq \| a\nabla e \|_{L^2(\Omega)} \cdot \| \nabla \tilde{e} \|_{L^2(\Omega)}$$

$$\leq c_1 \| e \|_{H_0^1(\Omega)} \cdot \| \tilde{e} \|_{H_0^1(\Omega)}$$

from Schwarz's inequality.

Young's inequality states $\| f \| \cdot \| g \| \leq \varepsilon \| f \|^2 + \frac{1}{4\varepsilon} \| g \|^2$. Using this inequality we get

$$|<a\nabla e, \nabla \tilde{e}>| \leq \varepsilon c_1 \| e \|^2_{H_0^1(\Omega)} + \frac{c_1}{4\varepsilon} \| \tilde{e} \|^2_{H_0^1(\Omega)}$$

Thus

$$\frac{1}{2} \frac{d}{dt} \| e \|^2_{L^2(\Omega)} + <\frac{\partial e}{\partial t} , \tilde{e}> + c_0 \| e \|^2_{H_0^1(\Omega)}$$

$$\leq \varepsilon c_1 \| e \|^2_{H_0^1(\Omega)} + \frac{c_1}{4\varepsilon} \| \tilde{e} \|^2_{H_0^1(\Omega)} \quad t \in (0, T]$$

and

$$\frac{1}{2}\frac{d}{dt}\|e\|^2_{L^2(\Omega)} + \langle\frac{\partial e}{\partial t},\tilde{e}\rangle + (c_0 - \varepsilon c_1)\|e\|^2_{H_0^1(\Omega)}$$

$$\leq \frac{c_1}{4\varepsilon}\|\tilde{e}\|^2_{H_0^1(\Omega)}, \qquad t \in (0,T]$$

Let $\varepsilon = c_0/2c_1$, then for $C = c_1^2/c_o$

$$\frac{d}{dt}\|e\|^2_{L^2(\Omega)} + 2\langle\frac{\partial e}{\partial t},\tilde{e}\rangle + c_0\|e\|^2_{H_0^1(\Omega)}$$

$$\leq C\|e\|^2_{H_0^1(\Omega)} .$$

Integrating from 0 to τ

$$\|e\|^2_{L^2(\Omega)}(\tau) - \|e\|^2_{L^2(\Omega)}(0) + c_0\int_0^\tau \|e\|^2_{H_0^1(\Omega)}\,dt$$

$$\leq C\int_0^\tau \|\tilde{e}\|^2_{H_0^1(\Omega)}\,dt - 2\int_0^\tau \langle\frac{\partial e}{\partial t},\tilde{e}\rangle\,dt$$

and

$$\int_0^\tau \langle\frac{\partial e}{\partial t},\tilde{e}\rangle\,dt = \langle e,\tilde{e}\rangle\Big|_0^\tau - \int_0^\tau \langle e, \frac{\partial\tilde{e}}{\partial t}\rangle\,dt$$

$$|\langle e,\tilde{e}\rangle| \leq \varepsilon\|e\|^2_{L^2(\Omega)} + C\|\tilde{e}\|^2_{L^2(\Omega)}$$

and

$$\left|\int_0^\tau \langle e, \frac{\partial e}{\partial t}\rangle\right| dt \leq \frac{1}{2}\left(\int_0^\tau \|e\|^2_{L^2(\Omega)}(t)\, dt + \int_0^\tau \|\frac{\partial \tilde{e}}{\partial t}\|^2_{L^2(\Omega)}(t)\, dt\right).$$

It follows that

$$\|e\|^2_{L^2(\Omega)}(\tau) - \|e\|^2_{L^2(\Omega)}(0) + c_0 \int_0^\tau \|e\|^2_{H_0^1(\Omega)}\, dt$$

$$\leq \|e\|^2_{L^2(\Omega)}(\tau) + C\Big(\|e\|^2_{L^2(\Omega)}(0) + \int_0^\tau \|\tilde{e}\|^2_{H_0^1(\Omega)}\, dt$$

$$+ \int_0^\tau \|e\|^3_{L^2(\Omega)}\, dt + \|\tilde{e}\|^2_{L^2(\Omega)}(\tau) + \|\tilde{e}\|^2_{L^2(\Omega)}(0)$$

$$+ \int_0^\tau \|\frac{\partial e}{\partial t}\|^2_{L^2(\Omega)}\, dt\Big)$$

Thus for ε sufficiently small

$$\|e\|^2_{L^2(\Omega)}(\tau) + \vartheta \cdot \int_0^\tau \|e\|^2_{H_0^1(\Omega)}\, dt$$

$$\leq C\Big(\int_0^\tau \|e\|^2_{L^2(\Omega)}\, dt + \|e\|^2_{L^2(\Omega)}(0) + \int_0^\tau \|\tilde{e}\|^2_{H_0^1(\Omega)}\, dt$$

$$+ \|\tilde{e}\|^2_{L^2(\Omega)}(\tau) + \|\tilde{e}\|^2_{L^2(\Omega)}(0) + \int_0^\tau \|\frac{\partial \tilde{e}}{\partial t}\|^2_{L^2(\Omega)}\, dt\Big)$$

Gronwall's inequality: Let f,g, and h be piecewise continuous nonnegative functions defined on an interval $a \leq t \leq b$ and assume that for each $t \in [a,b]$

$$f(t) + h(t) \leq g(t) + \int_0^\tau f(s)\, ds$$

Then

$$f(t) + h(t) \leq \int_a^r e^{t-s} g(s)\, ds + g(t)$$

and if g is nondecreasing

$$f(t) + h(t) \leq e^{t-a}\, g(t) .$$

Thus

$$\|e\|^2_{L^2(\Omega)}(\tau) + \partial \int_0^\tau \|e\|^2_{H_0^1(\Omega)}\, dt$$

$$\leq C \Big(\max_{0\leq t\leq T} \|e\|^2_{L^2(\Omega)}(t) + \int_0^\tau \|\tilde{e}\|^2_{H_0^1(\Omega)}\, dt$$

$$+ \int_0^\tau \|\frac{\partial \tilde{e}}{\partial t}\|^2_{L^2(\Omega)}\, dt + \|\tilde{e}\|^2_{L^2(\Omega)}(0) \Big)$$

Now by the definition of U(x,0)

$$\|u-U\|_{L^2(\Omega)}(0) \leq \|u-\tilde{u}\|_{L^2(\Omega)}(0)$$

and the desired result is obtained, that is

$$\max_{0\leq t\leq T} \|u-U\|^2_{L^2(\Omega)}(t) + \partial \int_0^\tau \|u-U\|^2_{H_0^1(\Omega)}\, dt$$

$$C \Big(\max_{0\leq t\leq T} \|u-\tilde{u}\|^2_{L^2(\Omega)}(t) + \int_0^\tau \|u-\tilde{u}\|^2_{H_0^1(\Omega)}(t)\, dt$$

$$+ \int_0^\tau \|\frac{\partial}{\partial t}(u-\tilde{u})\|^2_{L^2(\Omega)}(t)\, dt \Big)$$

This implies that the error bounds are dependent on how well u and its first partial derivative with respect to time can be approximated by the arbitrary map $\tilde{u}$. For Hermite cubics, $\| u-u \|_{L^2(\Omega)}$ is $O(h^4)$, and $\|u-\tilde{u}\|_{H_0^1(\Omega)}$ is $O(h^3)$. With further restrictions it can also be shown that the $L^2(\Omega)$ error is $O(h^4)$.

In [3] J. E. Dendy, Jr. proves the following theorem which gives error estimates for the discrete time case.

Theorem 2. Let u be the solution of

$$\left\langle \frac{\partial u}{\partial t}, v \right\rangle + \langle a(\cdot,u)\nabla u, \nabla v \rangle = \langle f(\cdot,u), v \rangle, \quad v \in H_0^1(\Omega)$$

$$u(x,0) = u_0(x) \qquad x \in \Omega$$

$$u(\cdot,t) \quad H_0^1(\Omega)^1, \qquad t \in (0,T)$$

and assume that $\lambda > \frac{1}{4} c_1$, $c_1 = \max a(x,t,u), (x,t) \in \overline{\Omega} \times (0,t]$. Let $E^m = U^m - W^m$, where $U^m \in M$, and M satisfies $\inf \|u-\tilde{u}\|_{H^3(\Omega)} \leq C h^{p-s} \|u\|_{H^r(\Omega)}$ [3]. Let W be the mapping from: $(0,T] \to M$ defined by $\langle a(\cdot,u\ (u-W),V \rangle + \langle u-W,V \rangle = 0$ for all $V \in M$. If $u \in C^4(\overline{\Omega} \times (0,T])$, and $u, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2}$ $L^2(0,T;H^r(\Omega))$, $r \geq 2$, then for $1 < N < M$ and h and Δt sufficiently small

$$\partial \sum_{m=1}^{N-1} \| \partial E^m \|^2 \Delta t + q[\|E^N\|_1^2 + \|E^{N-1}\|_1^2]$$

$$+ 2\lambda^2 \Delta t \, \| \frac{\partial^2 m}{\partial x \partial y} (E^N - E^{N-1}) \|^2 \leq e^{c\Delta t N} (\Delta t)^4$$

$$\| u^N - U^N) \| = e^{cN\Delta t} (\Delta t)^2 = c \, e^{cN\Delta t} \, h^r \quad ,$$

if

$$\|E^1\|_1^2 + \|E^0\|_1^2 + \Delta t \, \| \frac{\partial^2}{\partial x \partial y} (E^1 - E_0) \|^2 \leq c(\Delta t)^4$$

For the case of Hermite cubics, r = 4, and the conditions on M are satisfied. U is the approximate solution obtained in the Laplace modified alternating direction method.

Using the program it can be shown that the order of the error is actually this by approximating the value of ω where the error is $O(h^\omega)$. This approximation can be obtained in the following way:

Let e_1 be the error for $h = h_1$, and e_2 be the error for $h = h_2$, then

$$e_1 = C \, h_1^\omega \qquad\qquad e_2 = C \, h_2^\omega$$

$$\log(e_1/e_2) = \omega \log(h_1/h_2)$$

$$\omega = (\log(e_1) - \log(e_2))/(\log(h_1) - \log(h_2))$$

Using this method and choosing $t = h^{1/2}$ we obtain the results presented in Table 1. Table 2 shows similar results for the time error.

To obtain the estimates in the two tables the following problem was used for easily obtained analytic solution.

$$u_t = u_{xx} + u_{yy} - 2(e^{-t})((x-x^2) + (y-y^2) + \frac{1}{2}(x-x^2)(y-y^2))$$

$$u(\cdot,t) = 0 \qquad \text{for all } x \in \partial\Omega$$

$$u(x,y,0) = (x-x^2)(y-y^2)$$

In this test $\lambda = \frac{1}{4}$ was used in the discrete time formulation.

TABLE 1

H	DT	L-2 NORM	MAX NORM
0.2000	0.0400	0.0003680	0.0006815
0.1000	0.0100	0.0000286	0.0000619

THE VALUE OF W FOR THE MAX NORM IS 3.46006.
THE VALUE OF W FOR THE L-2 NORM IS 3.68510.

TABLE 2

H	DT	L-2 NORM	MAX NORM
0.2000	0.0400	0.0003680	0.0006815
0.1000	0.0100	0.0000286	0.0000619

THE VALUE OF W FOR THE MAX NORM IS 1.73003.
THE VALUE OF W FOR THE L-2 NORM IS 1.84255.

VI. TEST PROBLEMS

The utility of the program is evident in several fields of engineering. The following test problems will illustrate its use in two of these. The problems are taken from the fields of fluid dynamics and heat transfer.

A. RAYLEIGH'S PROBLEM FOR A CORNER

In the first problem we consider the diffusion equation in two dimensions with the following boundary and initial conditions

$$u_t = (1/Re)(u_{xx} + u_{yy}) \quad , \quad x,y > 0$$

$$u(x,y,0) = 1$$

$$u(0,y,t) = u(x,0,t) = 0$$

$$u(x,y,t) \to 1 \quad \text{as} \quad x \text{ and } y \to \infty$$

where Re is the Reynold's number of the fluid under consideration.

This is known as Rayleigh's problem for a corner, and the solution describes the impulsive motion of a right angled corner formed by two infinite flat plates and is used to infer the steady flow along the corner, with leading edge at t=0.

For the purpose of numerical approximations, using a Reynold's number of 1000, $x = 4$ and $y = 4$ can be considered as infinity and the problem can be stated for use in the program as

$$u_t = (1/1000)(u_{xx} + u_{yy})$$

$$u(x,y,0) = 1$$

$$u(0,y,t) = u(x,0,t) = 0$$

$$u(4,y,t) = u(x,4,t) = 1$$

and

$$a(x,y,t,u) = 1/1000$$

$$f(x,y,t,u,u_x,u_y) = 0$$

A nonuniform grid with a fine mesh near the $x = 0$, $y = 0$ corner is useful for an accurate description of the boundary layer behavior near the corner. Illustration (1) is a diagram of the grid used.

The exact solution to this problem can be found analytically to be

$$u(x,y,t) = \text{erf}(X)\ \text{erf}(Y)$$

where

$$X = (x/2)(Re/t)^{1/2}$$

$$Y = (y/2)(Re/t)^{1/2}$$

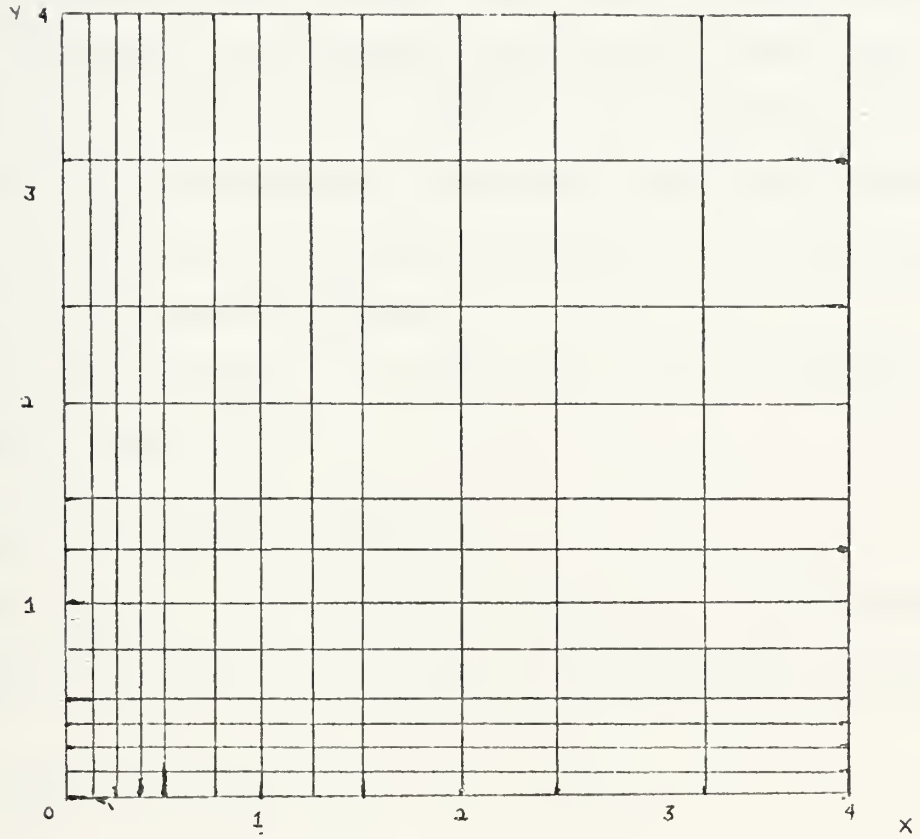


Illustration (1)

The nature of the solution is that of a shock wave traveling through the rectangle. The error in the approximation seems to be greatest on the wave and small even when the grid size is quite large. The error was also larger at early times near the boundary discontinuity than later in the process. Table 3 is a sample of the output of the program for this problem. The output points were taken near the discontinuity on the boundary since it is this area which is of greatest interest.

For this problem $\lambda = .00025$, $\Delta t = .03$, and the largest Δx was .75.

B. THE HEAT TRANSFER PROBLEM

The equation for heat conductance for heterogeneous isotropic solids and frictionless incompressible fluids is

$$\rho c \frac{\partial u}{\partial t} = \nabla \cdot (k\ u) + q_0'''$$

where the solution $u(x,y,t)$ represents the temperature of the material. ρ is the density of the material, c is the heat capacitance, k is the thermoconductivity, and q_0''' is an internal generation term. ρc is a constant and can be combined with k and q_0''' to give

$$\frac{\partial u}{\partial t} = \nabla \cdot (k'\ u) + q'''/\rho c$$

In this form k' is the thermal diffusitivity.

	X = 0.01409 Y = 0.01409		X = 0.06250 Y = 0.06250		X = 0.11091 Y = 0.11091		X = 0.13909 Y = 0.13909	
T	U	EXACT U	U	EXACT U	U	EXACT U	U	EXACT U
0.900	0.0745918	0.0676777	0.7196220	0.7383735	1.0138583	0.9821940	0.9864668	0.9979126
1.800	0.0364966	0.0344596	0.4818977	0.4934148	0.8940369	0.8751153	0.9584748	0.9595337
2.700	0.0240360	0.0231134	0.3602012	0.3659813	0.7634104	0.7547879	0.8869747	0.8866272
3.600	0.0178927	0.0173880	0.2869043	0.2901084	0.6585829	0.6542013	0.8083950	0.8078806
4.500	0.0142427	0.0139359	0.2381321	0.2400827	0.5765228	0.5740266	0.7356380	0.7351071
5.400	0.0118277	0.0116274	0.2034176	0.2046949	0.5115582	0.5100022	0.6716475	0.6711319
6.300	0.0101128	0.0099751	0.1774808	0.1783655	0.4592197	0.4581812	0.6162240	0.6157337
7.200	0.0088325	0.0087339	0.1573808	0.1580209	0.4163111	0.4155806	0.5683175	0.5678706
8.100	0.0078402	0.0077675	0.1413537	0.1418334	0.3805711	0.3800362	0.5267705	0.5263647

TABLE 3

For most materials the thermal diffusitivity can be approximated by a linear function of the temperature. It is typically of the form

$$k' = k + \alpha(u - u_0)$$

where k is the initial value of the thermal diffusitivity and α is a small real number.

The internal generation term is highly variable and dependent on the type of problem. A typical function encountered is of the form

$$e^{\beta(u/u_0)}$$

Analytic solutions for problems such as this are not available and it is in these cases that approximation techniques are invaluable.

The problem for the program run will be as follows

$$u_t = \nabla\cdot((100 + .005(u-u_0))\nabla u) + e^{(u/u_0)}$$

$$u(x,y,0) = \sin(\pi x) + \sin(\pi y)$$

$$u(0,y,t) = u(1,y,t) = e^{(-\pi^2 t)} \sin(\pi y)$$

$$u(x,0,t) = u(x,1,t) = e^{(-\pi^2 t)} \sin(\pi x)$$

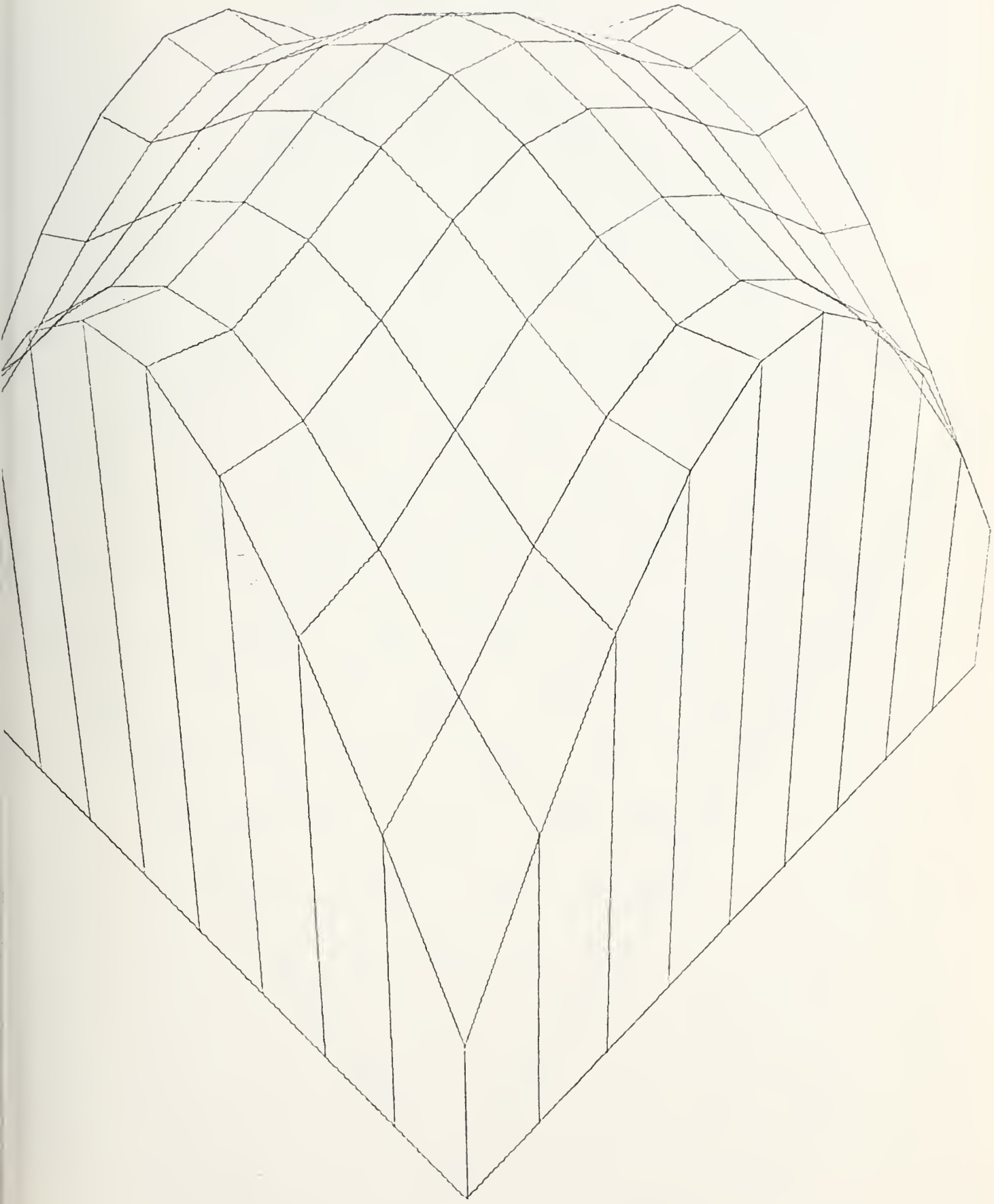
$$a(x,y,t,u) = 100 + .005(u - (\sin(\pi x) + \sin(\pi y)))$$

$$f(x,y,t,u,u_x,u_y) = e^{(u/(\sin(\pi x)+\sin(\pi y)))}$$

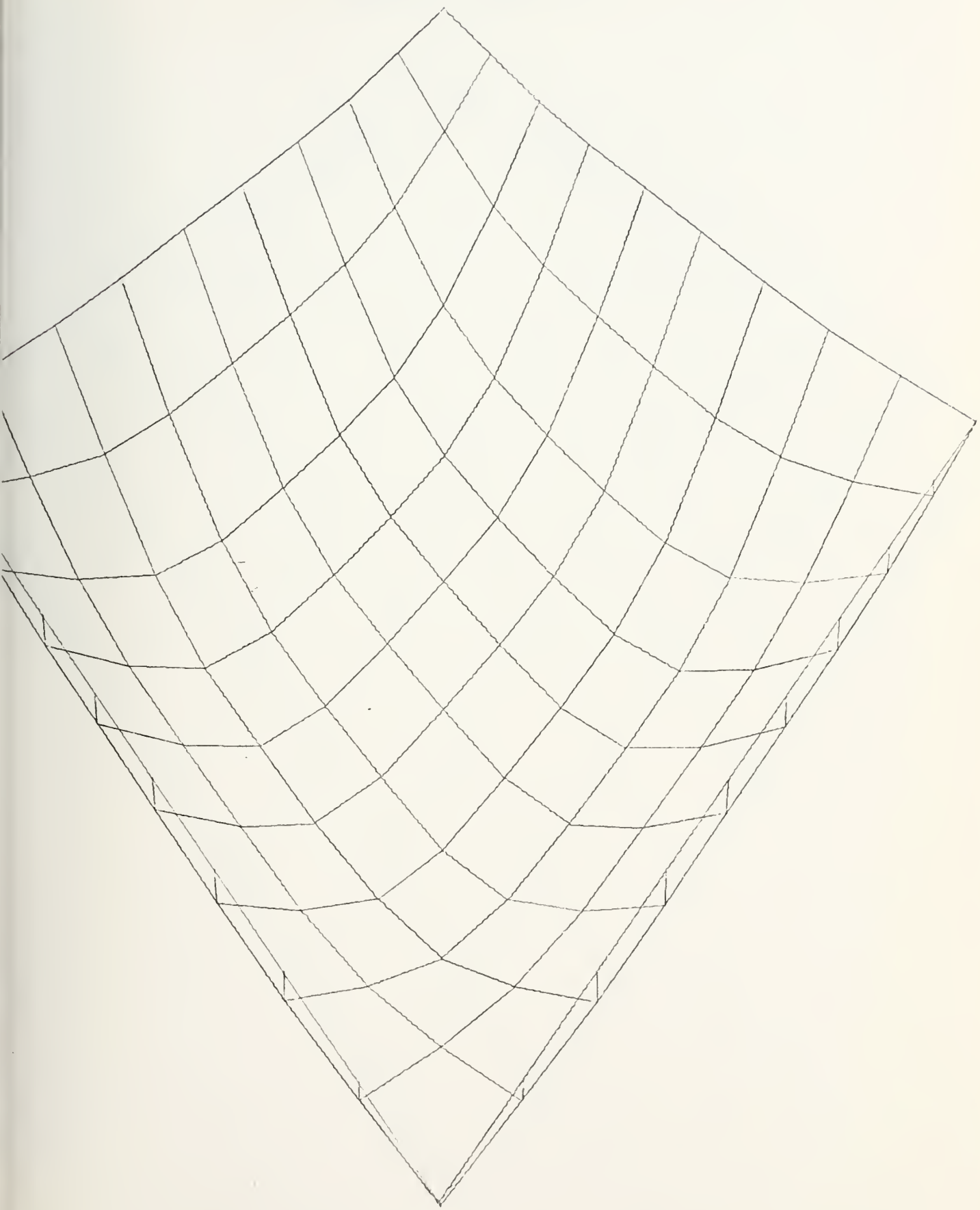
with uniform grid, with $\Delta x = \Delta y = h = 1/10$, time step $\Delta t = .01$ and $\lambda = 30$.

Plots 1-8 show the general nature of the solution. Plot 1 shows the solution at an early time while it retains much of its initial form. The area near the boundary is beginning to become smaller. The internal generation term causes the oscillatory nature of the solution. After a longer period of time the solution begins to take on a steady state of zero.

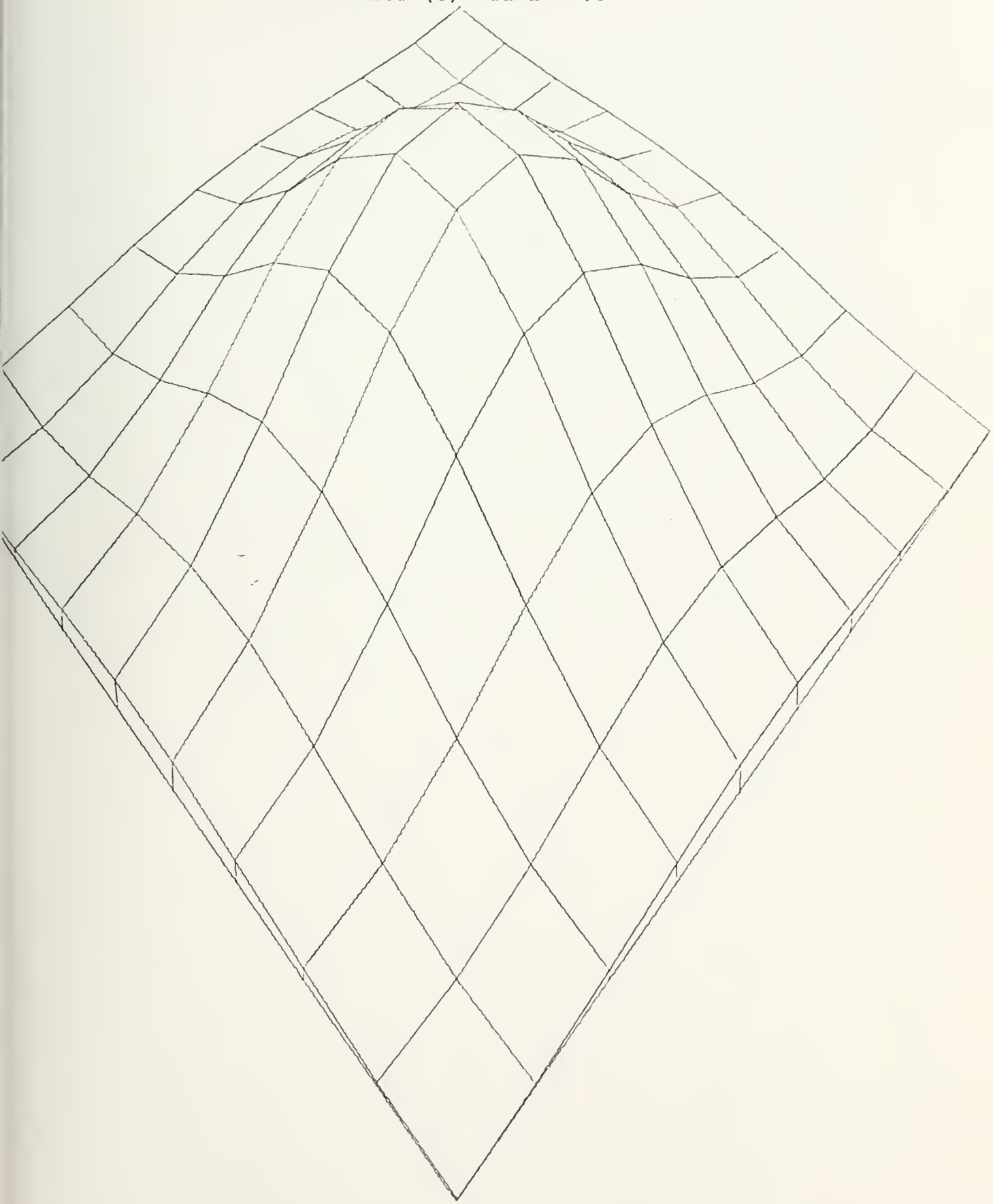
PLOT (1) TIME = .05



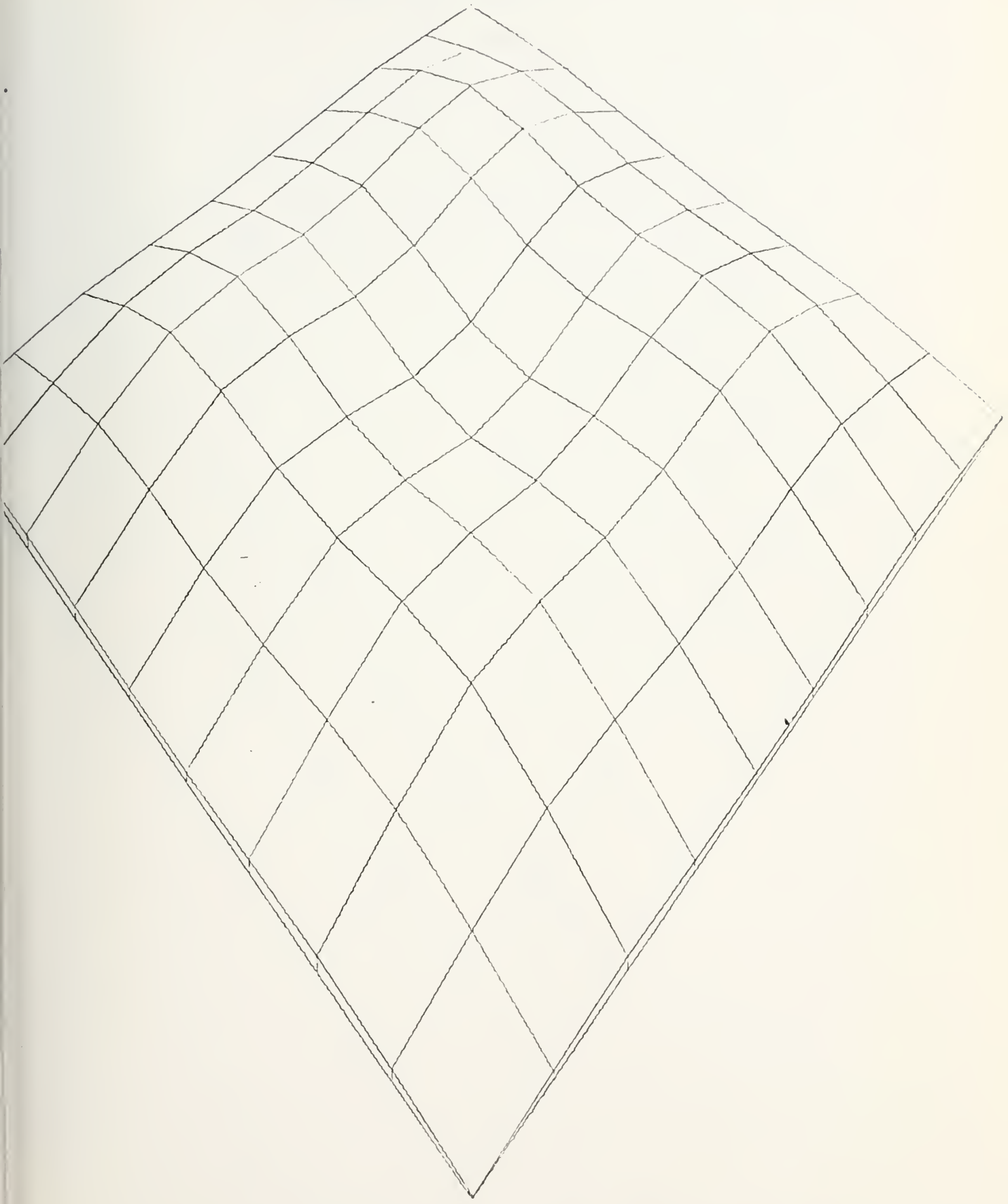
PLOT (2) TIME = .40



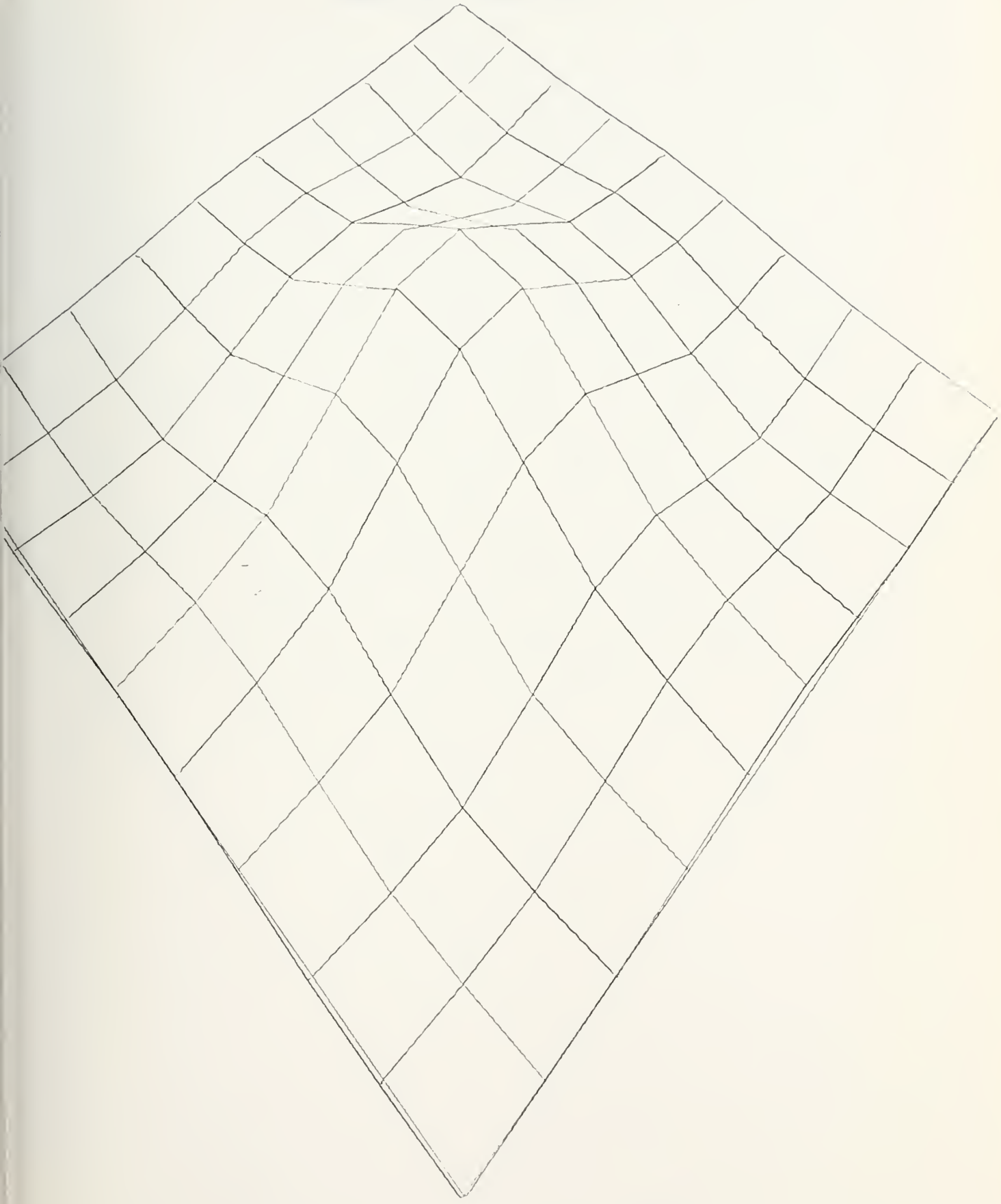
PLOT (3) TIME = .5



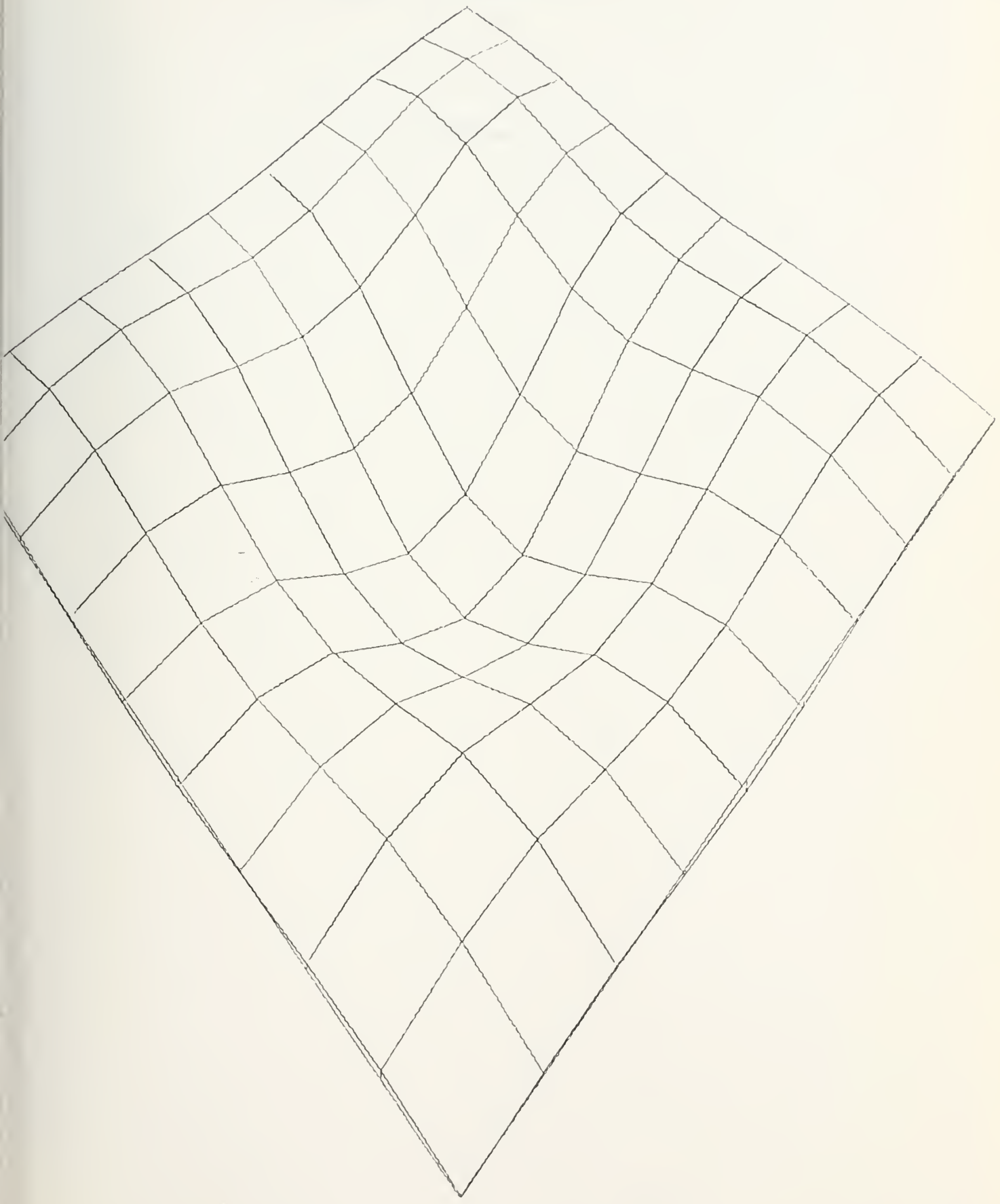
PLOT (4) TIME = .6



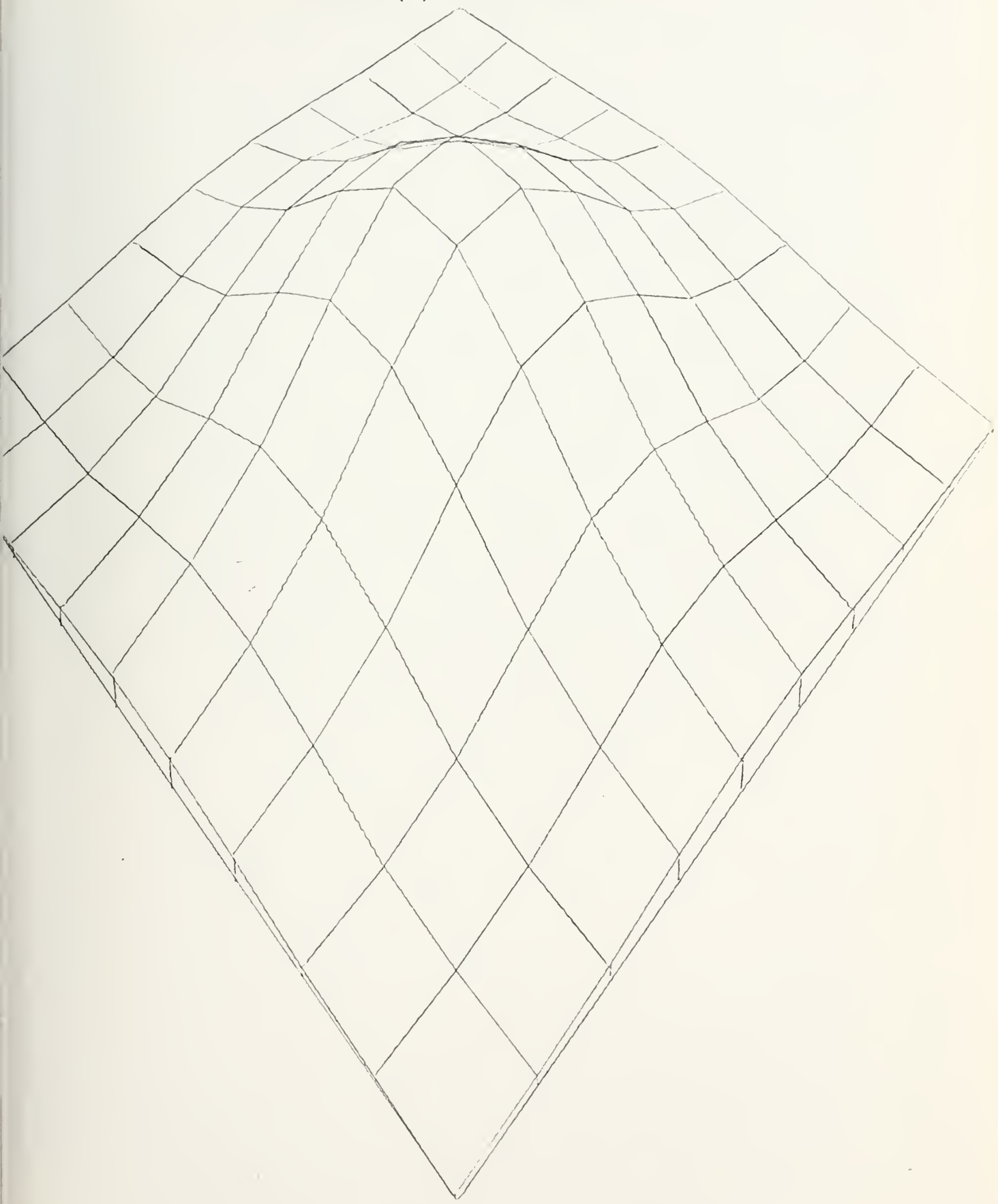
PLOT (5) TIME = .7



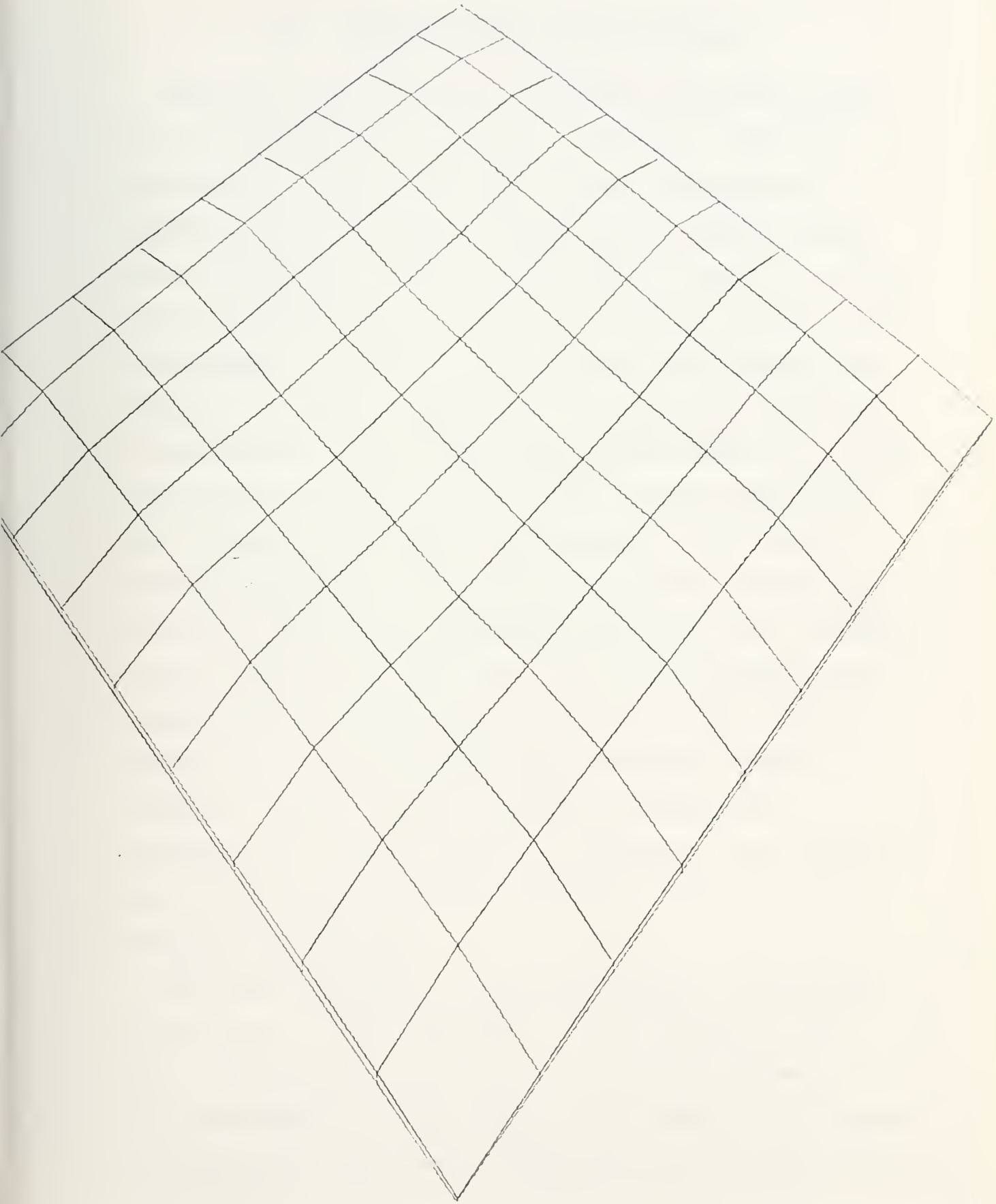
PLOT (6) TIME = .9



PLOT (7) TIME = .11



PLOT (8) TIME = 2.55



VII. CONCLUSIONS AND RECOMMENDATIONS

Two major conclusions can be reached from the application of the program to the various problems. First the program is in general more efficient than the standard implementation of Galerkin methods which achieve the same degree of accuracy. It is also possible to achieve a higher degree of continuity and approximation in the solution using higher degree B-splines. It is possible that changes not evident to the author may be made to improve the efficiency of the program. Most of the computational effort in a given time step is in evaluating U^n and ∇U^n at the gauss points; more efficient evaluations or placement of the quadrature points can possibly increase the speed of the program. It should be noted that the matrices [see (3.8),(3.9)] arising need be factored only once since they are time-independent. Second, optimal error bounds for the method have not been derived in the case that the Laplace modified forward difference method is used to obtain the second set of coefficients so that the centered difference method may be used, but this does not seem to have any effect on the solution in application.

Using the program as a basis the Galerkin methods may be extended to a third dimension, using extension of the alternating direction to obtain it. The basic ideas used in the development of the program can be utilized to program the second order hyperbolic problem using the Galerkin methods.

APPENDIX A

THE COMPUTER PROGRAM

A. THE PROGRAM VARIABLES

Alpham - the coefficients of the splines at time m.

CX,CY,AX,AY - the matrices C_x, C_y, A_x, A_y respectively described in Section III

X,Y - the grid points on the x and y axes

TKX,TKY - the knots system on the x and y axes, used for the de Boor routines

GX,GY - the Gauss weights computed in WEIGHT

EKCX,ETAY - the values of the basis functions at the quadrature points

ALXR,ALXL,ALYB,ALYT - the coefficients for the projection of the boundary conditions at time m

CPX,CPY - the quadrature points on the x and y axes

NX,NY - the number of intervals on the x and y axes

IAX,IAY - the dimension of the matrices defined above

DT - Δt

TIME - time m

B. THE MAIN PROGRAM

The main program calls various subroutines to accomplish the necessary procedures. Brief indications of the function of each subroutine will follow. The program begins by setting up the grid desired and its corresponding knots system and then computes the necessary matrices. It stores

parts of the matrices which will be changed but needed for future corrections. The first set of coefficients is obtained by projecting the initial conditions into the space. The second set is computed using a forward difference Laplace modified step. The program then steps forward in time using the central difference Laplace modified method.

C. MATCOM

MATCOM computes the matrices for the program and stores them in band symmetric storage used by IMSL library routines. It also stores the quadrature points and the values of the basis functions at the quadrature points.

D. FMATCO

FMATCO projects the initial conditions into the space.

E. WEIGHT

WEIGHT computes and stores the product of the quadrature weights and Δx and Δy for the appropriate interval.

F. GRID

GRID computes the grid points.

G. KNOTS

KNOTS computes the knot system for the de Boor routines.

H. UEVAL

UEVAL computes the value of the function at specific points (because of inefficiency its use should be limited).

I. MULTIP

MULTIP computes (CX CY) e^m in manner described in Section III.

J. SOLVER

SOLVER solves system of the form (CX I)(I CY) $e^m = \beta$ in the manner described in Section III.

K. ANEVAL

ANEVAL evaluates the function and its first partial derivatives at the quadrature points.

L. RTHDSD

RTHDSD computes the right hand side of equation (3.8).

M. ALXYBC

ALXYBC projects the boundary conditions into the space.

N. FUNCTION F

Function F defines the necessary functions for the problem being solved.

O. THE DE BOOR SUBROUTINES

The de Boor subroutines perform various operations to obtain B-splines [2].

APPENDIX B

THE L^2 PROJECTION OF FUNCTIONS

The L^2 projection of the initial conditions and other functions involved in the program were accomplished using Gauss quadrature. Appendix A describes several subroutines in which this is done. To achieve the desired accuracy it was necessary to use nine quadrature points for each interval. In the subroutine WEIGHT, the Gauss weights are combined with the value of Δx and Δy for their respective intervals and saved. This is necessary in the case of non-uniform grids. The quadrature points are stored for future use in the subroutine MATCOM.

To project a function into the space, we must take its inner produce with each of the basis functions. This means we must know the values of the basis functions at the quadrature points. These are computed using the de Boor routines and saved in MATCOM. In the case of one dimensional B-splines there are only four non-zero basis functions on each interval and each basis function is non-zero on no more than two intervals. Thus on each interval the quadrature is taken with the functions and each of the four non-zero basis functions.

For example, on the I^{th} x-interval and the J^{th} y-interval, we evaluate the function at the nine quadrature points and take its product with the value of the basis functions at

the point and the appropriate Gauss weights, and then sum over the nine points in the interval for each of the four basis functions. This procedure is continued by moving to the next interval, and using the appropriate basis functions for that interval.

LIST OF PROGRAM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                          C
C                                                                          C
C    THE FIRST SECTION OF THE MAIN PROGRAM COMPUTES THE GRID POINTS, THE   C
C    KNOT SYSTEM,AND THE COEFFICIENT MATRICES.  IT STORES PARTS OF THE     C
C    MATRICES WHICH WILL LATER BE DESTROYED.  THE MATRICES ARE REDUCED TO
C    THE FORM NECESSARY FOR THE LINEAR SYSTEM SOLVER.                      C
C                                                                          C
C                                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION ALPHAM(1800),ALPHA(1800),BMP(1800),ULX(50,4),ULY(50,4)
      DIMENSION X(25),Y(25),TKX(50),TKY(50),CX(50,4),CY(50,4),BETA(1800)
      DIMENSION EM(1800),AX(50,4),AY(50,4),ULCX(50,4),ULCY(50,4)
      DIMENSION CXS(4,2),CYS(4,2),AXS(4,2),AYS(4,2)
      COMMON GX(60),GY(60),EKCX(60,4,2),ETAY(60,4,2),ALXR(50),ALXL(50),
     CALYT(50),ALYB(50),CPX(60),CPY(60),IAX,IAY,NX,NY,TIME,DT
      DATA IDGT,NC,K/10,3,4/
      DATA XL,XR,NUX,YB,YT,NUY/0.,1.,2,0.,1.,2/
  379 TIME=0.
      NX=12
      NY=12
      NXP=NX+1
      NYP=NY+1
      DT=.009
      ELAM=.00025
      ICOUN=0
      TLDT=2.*ELAM*DT
      CALL GRID(NX,XL,XR,X)
      CALL GRID(NY,YB,YT,Y)
      CALL WEIGHT(X,Y,NX,NY)
      CALL KNOTS(X,TKX,NX,NUX,NDX,K)
      CALL KNOTS(Y,TKY,NY,NUY,NDY,K)
      CALL MATCOM(TKX,K,NDX,CX,AX,CPX,IAX,EKCX,GX)
      CALL MATCOM(TKY,K,NDY,CY,AY,CPY,IAY,ETAY,GY)
      DO 3 I=1,4
      CXS(I,1)=CX(I,5-I)
      CYS(I,1)=CY(I,5-I)
      CX(I,5-I)=0.0
      CY(I,5-I)=0.0
      AXS(I,1)=AX(I,5-I)
      AYS(I,1)=AY(I,5-I)
      AX(I,5-I)=0.0
      AY(I,5-I)=0.0
      CXS(5-I,2)=CX(IAX,I)
      CYS(5-I,2)=CY(IAY,I)
      CX(IAX,I)=0.0
      CY(IAY,I)=0.0
      AXS(5-I,2)=AX(IAX,I)
```

```
      AYS(5-I,2)=AY(IAY,I)
      AX(IAX,I)=0.0
      AY(IAY,I)=0.0
    3 CONTINUE
      DO 101 I=1,4
      DO 101 J=1,2
      AXS(I,J)=CXS(I,J)+TLDT*AXS(I,J)
      AYS(I,J)=CYS(I,J)+TLDT*AYS(I,J)
  101 CONTINUE
      CX(1,4)=1.0
      CX(IAX,4)=1.0
      CY(1,4)=1.0
      CY(IAY,4)=1.0
      CALL LUDAPB(CX,IAX,NC,50,ULCX,50,D1,D2,IER)
      CALL LUDAPB(CY,IAY,NC,50,ULCY,50,D1,D2,IER)
      DO 4 J=1,4
      DO 5 I=1,IAX
      AX(I,J)=CX(I,J)+TLDT*AX(I,J)
    5 CONTINUE
      DO 4 I=1,IAY
      AY(I,J)=CY(I,J)+TLDT*AY(I,J)
    4 CONTINUE
      CALL LUDAPB(AX,IAX,NC,50,ULX,50,D1,D2,IER)
      CALL LUDAPB(AY,IAY,NC,50,ULY,50,D1,D2,IER)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    AT THIS POINT THE INITIAL CONDITIONS ARE PROJECTED INTO THE SPACE AND C
C    THE FIRST SET OF COEFFICIENTS ARE OBTAINED.                        C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        CALL FMATCO(ALPHAM)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    USING THE LAPLACE-MODIFIED FORWARD DIFFERENCE SCHEME, THE FIRST STEP C
C    AHEAD IN TIME IS MADE AND THE SECOND SET OF COEFFICIENTS ARE OBTAINED. C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL ALXYBC(ULCX,ULCY,CXS,CYS)
      CALL SOLVER(ULCX,ULCY,ALPHAM,CXS,CYS,CY)
      CALL RTHDSD(TKX,TKY,ALPHAM,X,Y,EM)
      ITOT=IAX*IAY
      DO 12 I=1,ITOT
      EM(I)=EM(I)/2.
   12 CONTINUE
```

```
      TIME=TIME+DT
      ICOUN=ICOUN+1
      CALL ALXYBC(ULCX,ULCY,CXS,CYS)
      DO 72 I=1,IAX
      KM=(I-1)*IAY+1
      KMN=I*IAY
      ALXL(I)=ALXL(I)-(ALPHAM(KM))
      ALXR(I)=ALXR(I)-(ALPHAM(KMN))
   72 CONTINUE
      DO 73 J=1,IAY
      IC=(IAX-1)*IAY+J
      ALYT(J)=ALYT(J)-(ALPHAM(IC))
      ALYB(J)=ALYB(J)-(ALPHAM(J))
   73 CONTINUE
      CALL SOLVER(ULX,ULY,EM,AXS,AYS,AY)
      DO 74 I=1,ITOT
      ALPHAM(I)=ALPHAM(I)+EM(I)
   74 CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE PROGRAM NOW HAS THE TWO INITIAL SETS OF COEFFICIENTS NECESSARY TO   C
C     USE THE LAPLACE-MODIFIED CENTERED DIFFERENCE SCHEME.  TO STEP AHEAD IN  C
C     TIME THE LOOP SHOULD RETURN TO STATEMENT 999.                    C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  999 CALL       MULTIP(EM,BMP,CX,CY,IAX,IAY,CXS,CYS)
      CALL RTHDSD(TKX,TKY,ALPHAM,X,Y,BETA)
      DO 10 I=1,ITOT
      BETA(I)=BMP(I)+BETA(I)
   10 CONTINUE
      TIME=TIME+DT
      ICOUN=ICOUN+1
      CALL ALXYBC(ULCX,ULCY,CXS,CYS)
      DO 70 I=1,IAX
      KM=(I-1)*IAY+1
      KMN=I*IAY
      ALXL(I)=ALXL(I)-(ALPHAM(KM)+EM(KM))
      ALXR(I)=ALXR(I)-(ALPHAM(KMN)+EM(KMN))
   70 CONTINUE
      DO 71 J=1,IAY
      IC=(IAX-1)*IAY+J
      ALYT(J)=ALYT(J)-(ALPHAM(IC)+EM(IC))
      ALYB(J)=ALYB(J)-(ALPHAM(J)+EM(J))
   71 CONTINUE
      CALL SOLVER(ULX,ULY,BETA,AXS,AYS,AY)
      DO 15 I=1,ITOT
```

```
      ALPHAM(I)=BETA(I)+EM(I)+ALPHAM(I)
      EM(I)=BETA(I)+EM(I)
   15 CONTINUE
      STOP
      END
```

```
      SUBROUTINE MATCOM(T,K,NDIM,AM,S,CP,IA,EKC,G)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                                                                     C
C    THE SUBROUTINE MATCOM COMPUTES THE COEFFICIENT MATRICES FOR THE  C
C    PROGRAM.  IT ALSO COMPUTES AND STORES THE QUADRATURE POINTS AND THE
C    VALUES OF THE B-SPLINES AT THE QUADRATURE POINTS.                C
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION EKC(60,4,2),T(50),RHO(3),CP(60),AM(50,4),S(50,4),G(60)
      DIMENSION VNIKX(4,2)
      RHO(1)=-.7745966692
      RHO(2)=0.
      RHO(3)=-RHO(1)
      DO 10 I=1,NDIM
      DO 10 J=1,4
      AM(I,J)=0.0
      S(I,J)=0.0
   10 CONTINUE
      I=0
      M=0
      DO 80 ILEFT=K,NDIM,2
      M=M+1
      ILP1=ILEFT+1
      XM=T(ILP1)+T(ILEFT)
      DX=T(ILP1)-T(ILEFT)
      DO 30 ID=1,3
      I=I+1
      X=(XM+DX*RHO(ID))/2.0
      CP(I)=X
      CALL BSPLVD(T,K,X,ILEFT,VNIKX,2)
      DO 70 IBASE=1,K
      EKC(I,IBASE,1)=VNIKX(IBASE,1)
      EKC(I,IBASE,2)=VNIKX(IBASE,2)
      IA=IBASE+(M-1)*2
      DO 60 ICD=IBASE,K
      JA=ICD+(M-1)*2
      JB=5-((JA-IA)+1)
      IK=IA+4-JB
      S(IK,JB)=S(IK,JB)+G(I)*VNIKX(IBASE,2)*VNIKX(ICD,2)
      AM(IK,JB)=AM(IK,JB)+G(I)*VNIKX(IBASE,1)*VNIKX(ICD,1)
   60 CONTINUE
   70 CONTINUE
   30 CONTINUE
   80 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FMATCO(ALPHA)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                          C
C                                                                          C
C     THE SUBROUTINE FMATCO MAKES THE L-2 PROJECTION OF THE INITIAL CON-   C
C     DITIONS INTO THE SPACE.                                              C
C                                                                          C
C                                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION ALPHA(1800),ALPHAM(1800)
      COMMON GX(60),GY(60),EKCX(60,4,2),ETAY(60,4,2),AL(200),CPX(60),
     CCPY(60),IAX,IAY,NX,NY,TIME,DT
      DATA PI,PIS/3.141592654,9.869604404/
      IA=IAX*IAY
      DO 94 I=1,IA
      ALPHA(I)=0.0
   94 CONTINUE
      DO 95 I=1,NX
      DO 95 J=1,NY
      DO 95 IBASE=1,4
      DO 95 JBASE=1,4
      IB=(I-1)*2+IBASE
      JB=(J-1)*2+JBASE
      KM=(IB-1)*IAY+JB
      DO 95 ICDX=1,3
      DO 95 JCDY=1,3
      NQPX=ICDX+(I-1)*3
      NQPY=JCDY+(J-1)*3
      FNI=F(CPX(NQPX),CPY(NQPY),0.,0.,0.,0.,1)
      ALPHA(KM)=ALPHA(KM)+FNI*EKCX(NQPX,IBASE,1)*ETAY(NQPY,JBASE,1)*
     CGX(NQPX)*GY(NQPY)
   95 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WEIGHT(X,Y,NX,NY)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C    THE SUBROUTINE WEIGHT COMPUTES THE PRODUCT OF THE GAUSS QUADRATURE  C
C    WEIGHTS AND THE GRID SPACING FOR THEIR RESPECTIVE INTERVALS.        C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION GWT(3),X(25),Y(25)
      COMMON GX(60),GY(60)
      GWT(1)=5./9.
      GWT(2)=8./9.
      GWT(3)=GWT(1)
      DO 1 I=1,NX
      DX=(X(I+1)-X(I))/2.
      DO 1 J=1,3
      N=3*(I-1)+J
      GX(N)=GWT(J)*DX
    1 CONTINUE
      DO 2 I=1,NY
      DY=(Y(I+1)-Y(I))/2.
      DO 2 J=1,3
      N=3*(I-1)+J
      GY(N)=GWT(J)*DY
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GRID(N,A,B,X)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C                                                                              C
C     THE SUBROUTINE GRID COMPUTES THE GRID POINTS.                            C
C                                                                              C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION X(1)                                                    SGR00020
      H = (B-A)/FLOAT(N)                                                SGR00030
      X(1) = A                                                          SGR00040
      DO 1 I = 2,N                                                      SGR00050
    1 X(I) = A + FLOAT(I-1)*H                                           SGR00060
      X(N+1) = B                                                        SGR00070
      RETURN                                                            SGR00080
      END                                                               SGR00090
```

```
      SUBROUTINE KNOTS(X,T,N,NU,NDIM,K)                                 SKN00010
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE SUBROUTINE KNOTS COMPUTES THE KNOT SYSTEM FOR USE WITH THE DE BOOR    C
C     B-SPLINE PACKAGE.                                                 C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION X(1),T(1)                                               SKN00020
      NP1 = N+1                                                         SKN00030
      NDIM=0                                                            SKN00040
      DO 2 I = 1,NP1                                                    SKN00050
      KMNU = K-NU                                                       SKN00060
      IF (I .EQ. 1) KMNU = K                                            SKN00070
      IF (I.EQ.NP1) KMNU = K                                            SKN00080
      DO 1 J = 1,KMNU                                                   SKN00090
      NDIM = NDIM+1                                                     SKN00100
      T(NDIM) = X(I)                                                    SKN00110
 1    CONTINUE                                                          SKN00120
 2    CONTINUE                                                          SKN00130
      NDIM = NDIM-K
      RETURN                                                            SKN00140
      END                                                               SKN00150
```

```
      SUBROUTINE UEVAL(TKX,TKY,XQP,YQP,UVAL,IDX,IDY,ALPHA,X,Y,IAX,IAY)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE UEVAL EVALUATES THE FUNCTION U(X,Y,T) AT SPECIFIC  C
C     POINTS, USING THE PIECEWISE POLYNOMIAL REPRESENTATION.           C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION TKX(50),TKY(50),ALPHA(1800),X(25),Y(25),C(100)
      DIMENSION ALP(100),ALPN(100)
      K=4
      NRX=IAX
      NRY=IAY
      DO 71 IJ=1,IAX
      DO 70 J=1,IAY
      KM=(IJ-1)*IAY+J
      ALP(J)=ALPHA(KM)
   70 CONTINUE
      CALL BSPLPP(TKY,ALP,IAY,K,Y,C,LYI,NRY)
      CALL RESET I
      ALPN(IJ)=PPVALJ(Y,C,LYI,K,YQP,IDY)
   71 CONTINUE
      CALL BSPLPP(TKX,ALPN,IAX,K,X,C,LXI,NRX)
      CALL RESET I
      UVAL=PPVALU(X,C,LXI,K,XQP,IDX)
      RETURN
      END
```

```
      SUBROUTINE MULTIP(EM,BMP,CX,CY,IAX,IAY,CXS,CYS)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE SUBROUTINE MULTIP MAKES THE TENSOR PRODUCT MULTIPLICATION ON THE  C
C     RIGHT HAND SIDE OF THE EQUATION.                                  C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION CXS(4,2),CYS(4,2)
      DIMENSION EM(1800),BMP(1800),CX(50,4),CY(50,4),EMUL(50),EMULT(50)
      NC=3
      DO 6 I=1,IAX
      DO 7 J=1,IAY
      KM=(I-1)*IAY+J
      EMUL(J)=EM(KM)
    7 CONTINUE
      CALL VMULQF(CY,IAY,NC,50,EMUL,1,50,EMULT,50)
      DO 1 K=2,4
      J=IAY+1-K
      EMULT(K)=EMULT(K)+CYS(K,1)*EMUL(1)
      EMULT(J)=EMULT(J)+CYS(K,2)*EMUL(IAY)
    1 CONTINUE
      DO 6 J=1,IAY
      KM=(I-1)*IAY+J
      BMP(KM)=EMULT(J)
    6 CONTINUE
      DO 8 I=1,IAY
      DO 9 J=1,IAX
      KM=(J-1)*IAY+I
      EMUL(J)=BMP(KM)
    9 CONTINUE
      CALL VMULQF(CX,IAX,NC,50,EMUL,1,50,EMULT,50)
      DO 2 K=2,4
      J=IAX+1-K
      EMULT(K)=EMULT(K)+CXS(K,1)*EMUL(1)
      EMULT(J)=EMULT(J)+CXS(K,2)*EMUL(IAX)
    2 CONTINUE
      DO 8 J=1,IAX
      KM=(J-1)*IAY+I
      BMP(KM)=-2.*EMULT(J)
    8 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SOLVER(ULCX,ULCY,BETA,CXS,CYS,CY)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE SUBROUTINE SOLVER SOLVES THE SYSTEM OF LINEAR EQUATIONS IN ITS C
C     TENSOR PRODUCT FORM.                                              C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION CY(50,4),EMUL(50),EMULT(50)
      DIMENSION ULCX(50,4),ULCY(50,4),BETA(1800),BET(50),GAM(50)
      DIMENSION CXS(4,2),CYS(4,2)
      COMMON G(120),EK(60,4,2),ET(60,4,2),ALXR(50),ALXL(50),ALYT(50),
     CALYB(50),CPX(60),CPY(60),IAX,IAY
      IAXM=IAX-1
      IAYM=IAY-1
      CXS(1,1)=1.
      CXS(1,2)=1.
      CYS(1,1)=1.
      CYS(1,2)=1.
      CALL VMULQF(CY,IAY,3,50,ALYT,1,50,EMULT,50)
      CALL VMULQF(CY,IAY,3,50,ALYB,1,50,EMUL,50)
      DO 7 I=2,4
      JK=IAY+1-I
      EMUL(I)=EMUL(I)+CYS(I,1)*ALYB(1)
      EMUL(JK)=EMUL(JK)+CYS(I,2)*ALYB(IAY)
      EMULT(I)=EMJLT(I)+CYS(I,1)*ALYT(1)
      EMULT(JK)=EMULT(JK)+CYS(I,2)*ALYT(IAY)
    7 CONTINUE
      DO 9 I=1,IAY
      KD=(IAX-1)*IAY+I
      BETA(I)=EMUL(I)
      BETA(KD)=EMULT(I)
    9 CONTINUE
      DO 8 I=2,IAXM
      DO 8 J=2,4
      KM=(J-1)*IAY+I
      KD=(IAX-J)*IAY+I
      JK=IAX+1-J
      BETA(KM)=BETA(KM)-CXS(J,1)*EMUL(I)
      BETA(KD)=BETA(KD)-CXS(J,2)*EMJLT(I)
    8 CONTINUE
      DO 1 I=2,IAYM
      DO 2 J=1,IAX
      KM=(J-1)*IAY+I
      BET(J)=BETA(KM)
    2 CONTINUE
      CALL LUFLPB(ULCX,BET,IAX,3,50,GAM)
```

```
      DO 1 J=2,IAXM
      KM=(J-1)*IAY+I
      BETA(KM)=GAM(J)
    1 CONTINUE
      DO 3 I=2,IAXM
      DO 4 J=1,IAY
      KM=(I-1)*IAY+J
      BET(J)=BETA(KM)
    4 CONTINUE
      DO 6 J=2,4
      JK=IAY+1-J
      BET(J)=BET(J)-CYS(J,1)*ALYB(I)
      BET(JK)=BET(JK)-CYS(J,2)*ALYT(I)
    6 CONTINUE
      BET(1)=ALYB(I)
      BET(IAY)=ALYT(I)
      CALL LUELPB(ULCY,BET,IAY,3,50,GAM)
      DO 3 J=1,IAY
      KM=(I-1)*IAY+J
      BETA(KM)=GAM(J)
    3 CONTINUE
      DO 10 I=1,IAX
      KM=(I-1)*IAY+1
      KMN=I*IAY
      BETA(KM)=ALXL(I)
      BETA(KMN)=ALXR(I)
      DO 10 J=1,IAY
      IC=(IAX-1)*IAY+J
      BETA(J)=ALYB(J)
      BETA(IC)=ALYT(J)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ANEVAL(N,M,I,J,BETA,IAX,IAY,V,IDX,IDY)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C                                                                              C
C     THE SUBROUTINE ANEVAL EVALUATES THE FUNCTION AND ITS PARTIAL DERIVATIVES C
C     AT THE QUADRATURE POINTS.                                                C
C                                                                              C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION BETA(1800),BET(4)
      COMMON G(120),EKCX(60,4,2),ETAY(60,4,2)
      V=0.
      DO 1  JB=1,4
      BET(JB)=0.
    1 CONTINUE
      LY=2*J-1
      LX=2*I-1
      DO 2  JK=1,4
      DO 3  IK=1,4
      KM=((LX+(IK-1))-1)*IAY+JK+LY-1
      BET(JK)=BET(JK)+BETA(KM)*EKCX(N,IK,IDX)
    3 CONTINUE
      V=V+BET(JK)*ETAY(M,JK,IDY)
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RTHDSD(TKX,TKY,ALPHA,X,Y,BETA)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE RTHDSD COMPUTES THE RIGHT HAND SIDE OF THE EQUATION.  C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION TKX(50),TKY(50),ALPHA(1800),X(25),Y(25),BETA(1800)
      DIMENSION APHIM(50,50),FPHIM(50,50)
      COMMON GX(60),GY(60),EKCX(60,4,2),ETAY(60,4,2),AL(200),CPX(60),
     CCPY(60),IAX,IAY,NX,NY,TIME,DT
      DO 89 I=1,IAX
      DO 89 J=1,IAY
      APHIM(I,J)=0.0
      FPHIM(I,J)=0.0
   89 CONTINUE
      DO 90 I=1,NX
      DO 90 J=1,NY
      DO 90 ICDX=1,3
      NQPX=ICDX+(I-1)*3
      DO 90 JCDY=1,3
      NQPY=JCDY+(J-1)*3
      XQP=CPX(NQPX)
      YQP=CPY(NQPY)
       CALL  ANEVAL(NQPX,NQPY,I,J,ALPHA,IAX,IAY,UVAL,1,1)
       CALL  ANEVAL(NQPX,NQPY,I,J,ALPHA,IAX,IAY,UVALX,2,1)
       CALL  ANEVAL(NQPX,NQPY,I,J,ALPHA,IAX,IAY,UVALY,1,2)
      FMU=F(XQP,YQP,TIME,UVAL,UVALX,UVALY,3)
      AMU=F(XQP,YQP,TIME,UVAL,UVALX,UVALY,5)
      DO 90 IBASE=1,4
      DO 90 JBASE=1,4
      IB=(I-1)*2  +IBASE
      JB=(J-1)*2  +JBASE
      BFA=EKCX(NQPX,IBASE,2)*ETAY(NQPY,JBASE,1)*UVALX
      BFB=EKCX(NQPX,IBASE,1)*ETAY(NQPY,JBASE,2)*UVALY
      BFF=EKCX(NQPX,IBASE,1)*ETAY(NQPY,JBASE,1)
      APHIM(IB,JB)=APHIM(IB,JB)+AMU*(BFA+BFB)*GX(NQPX)*GY(NQPY)
      FPHIM(IB,JB)=FPHIM(IB,JB)+FMU*BFF*GX(NQPX)*GY(NQPY)
   90 CONTINUE
      TDT=2.0 * DT
      DO 91 I=1,IAX
      DO 91 J=1,IAY
      KM=(I-1)*IAY+J
      BETA(KM)=(-APHIM(I,J)+FPHIM(I,J))*TDT
   91 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ALXYBC(CX,CY,CXS,CYS)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE SUBROUTINE ALXYBC MAKES THE L-2 PROJECTION OF THE BOUNDARY CON-  C
C     DITIONS INTO THE SPACE.                                           C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION CX(50,4),CY(50,4),CXS(4,2),CYS(4,2),BET(50),GAM(50)
      COMMON GX(60),GY(60),EKCX(60,4,2),ETAY(60,4,2),ALXR(50),ALXL(50),
     CALYT(50),ALYB(50),CPX(60),CPY(60),IAX,IAY,NX,NY,TIME
      DATA PI,PIS/3.141592654,9.869604404/
      ALYT(IAY)=F(1.,1.,TIME,0.,0.,0.,9)
      ALYB(IAY)=F(0.,1.,TIME,0.,0.,0.,8)
      ALXL(IAX)=F(1.,0.,TIME,0.,0.,0.,6)
      ALXR(IAX)=F(1.,1.,TIME,0.,0.,0.,7)
      ALYT(1)=F(1.,0.,TIME,0.,0.,0.,9)
      ALYP(1)=F(0.,0.,TIME,0.,0.,0.,8)
      ALXL(1)=F(0.,0.,TIME,0.,0.,0.,6)
      ALXR(1)=F(0.,1.,TIME,0.,0.,0.,7)
      DO 55 I=1,IAX
      BET(I)=0.
      GAM(I)=0.
   55 CONTINUE
      DO 56 I=1,NX
      DO 56 IB=1,4
      IBA=(I-1)*2+IB
      DO 56 ICD=1,3
      N=(I-1)*3+ICD
      FNL=F(CPX(N),0.,TIME,0.,0.,0.,6)
      FNR=F(CPX(N),1.,TIME,0.,0.,0.,7)
      BET(IBA)=BET(IBA)+FNL*EKCX(N,IB,1)*GX(N)
      GAM(IBA)=GAM(IBA)+FNR*EKCX(N,IB,1)*GX(N)
   56 CONTINUE
      DO 58 I=2,4
      IK=IAX+1-I
      BET(I)=BET(I)-CXS(I,1)*ALXL(1)
      BET(IK)=BET(IK)-CXS(I,2)*ALXL(IAX)
      GAM(I)=GAM(I)-CXS(I,1)*ALXR(1)
      GAM(IK)=GAM(IK)-CXS(I,2)*ALXR(IAX)
   58 CONTINUE
      BET(1)=ALXL(1)
      GAM(1)=ALXR(1)
      BET(IAX)=ALXL(IAX)
      GAM(IAX)=ALXR(IAX)
      CALL LUFLPB(CX,BET,IAX,3,50,ALXL)
      CALL LUELPB(CX,GAM,IAX,3,50,ALXR)
```

```
      DO 54 I=1,IAY
      BET(I)=0.
      GAM(I)=0.
   54 CONTINUE
      DO 57 I=1,NY
      DO 57 IB=1,4
      IBA=(I-1)*2+IB
      DO 57 ICD=1,3
      N=(I-1)*3+ICD
      FNB=F(0.,CPY(N),TIME,0.,0.,0.,8)
      FNT=F(1.,CPY(N),TIME,0.,0.,0.,9)
      BET(IBA)=BET(IBA)+FNB*ETAY(N,IB,1)*GY(N)
      GAM(IBA)=GAM(IBA)+FNT*ETAY(N,IB,1)*GY(N)
   57 CONTINUE
      DO 59 I=2,4
      IK=IAY+1-I
      BET(I)=BET(I)-CYS(I,1)*ALYB(1)
      BET(IK)=BET(IK)-CYS(I,2)*ALYB(IAY)
      GAM(I)=GAM(I)-CYS(I,1)*ALYT(1)
      GAM(IK)=GAM(IK)-CYS(I,2)*ALYT(IAY)
   59 CONTINUE
      BET(1)=ALYB(1)
      GAM(1)=ALYT(1)
      BET(IAY)=ALYB(IAY)
      GAM(IAY)=ALYT(IAY)
      CALL LUELPB(CY,BET,IAY,3,50,ALYB)
      CALL LUELPB(CY,GAM,IAY,3,50,ALYT)
      ALYT(IAY)=F(1.,1.,TIME,0.,0.,0.,9)
      ALYB(IAY)=F(0.,1.,TIME,0.,0.,0.,8)
      ALXL(IAX)=F(1.,0.,TIME,0.,0.,0.,6)
      ALXR(IAX)=F(1.,1.,TIME,0.,0.,0.,7)
      ALYT(1)=F(1.,0.,TIME,0.,0.,0.,9)
      ALYB(1)=F(0.,0.,TIME,0.,0.,0.,8)
      ALXL(1)=F(0.,0.,TIME,0.,0.,0.,6)
      ALXR(1)=F(0.,1.,TIME,0.,0.,0.,7)
      RETURN
      END
```

```
      FUNCTION F(X,Y,T,U,UX,UY,K)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                                                                     C
C     THE FUNCTION SUBPROGRAM F LISTS THE FUNCTIONS FOR THE SPECIFIC PROBLEM.
C     F-1 IS THE INITIAL CONDITION FUNCTION.  F-3 IS THE FUNCTION F.  F-5 IS C
C     THE FUNCTION A.  F-6 THROUGH F-9 ARE THE BOUNDARY CONDITIONS.   C
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      GO TO (1,2,3,4,5,6,7,8,9),K
    1 F=1.
      RETURN
    2 F=0.
      RETURN
    3 F=0.
      RETURN
    4 F=0.
      RETURN
    5 F=.001
      RETURN
    6 F=0.
      RETURN
    7 F=1.
      RETURN
    8 F=0.
      RETURN
    9 F=1.
      RETURN
      END
```

LIST OF REFERENCES

1. J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, The Theory of Splines and Their Applications, Academic Press, New York and London, 1967.

2. C. R. de Boor, Package for Calculating with B-Splines, Mathematics Research Center, Madison, Wisconsin, 1973.

3. J. E. Dendy, Jr., An Analysis of Some Galerkin Schemes for the Solution of Nonlinear Time-Dependent Problems, Los Alamos Scientific Laboratory, Los Alamos, New Mexico, 1973.

4. J. E. Dendy, Jr., Galerkin's Method for Some Highly Non-Linear Problems, Los Alamos Scientific Laboratory, Los Alamos, New Mexico, 1973.

5. J. E. Dendy, Jr., and G. Fairweather, "Alternating-Direction Galerkin Methods for Parabolic and Hyperbolic Problems on Rectangular Polygons," SIAM J. Numer. Anal., 12 (1975), pp. 144-163.

6. J. Douglas, Jr., and T. Dupont, "Alternating Direction Galerkin Methods on Rectangles," Proc. Symposium on Numerical Solution of Partial Differential Equations, II, (B. Hubbard, ed.), pp. 133-214, Academic Press, New York, 1971.

7. G. Fairweather, A Survey of Discrete Galerkin Methods for Parabolic Equations in One Space Variable, Math. Colloq. U. C. T., 7 (1971-72), pp. 43-77.

8. G. Fairweather, Galerkin Methods for Differential Equations - Second edition, National Research Institute for Mathematical Sciences, Pretoria, S. Africa, 1973.

9. M. F. Wheeler, "A Priori L^2 Error Estimates for Galerkin Approximations to Parabolic Differential Equations, SIAM J. Numer. Anal., 10 (1973), pp. 723-759.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst. Prof. David A. Archer, Code 53Ac Department of Mathematics Naval Postgraduate School Monterey, California 93940	10
4.	Professor Craig Comstock, Code 53Zk Department of Mathematics Naval Postgraduate Shhool Monterey, California 93940	1
5.	Professor Frank D. Faulkner Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
6.	Asst. Prof. Richard Franke Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
7.	Professor L. D. Kovach, Code 53Kv Chairman, Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
8.	Professor R. Terry Williams, Code 51 Department of Meteorology Naval Postgraduate School Monterey, California 93940	1
9.	Assoc. Professor Thomas E. Cooper, Code 59Cg Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
10.	ENS. John Mark Franklin 2828 3rd Place East Tuscaloosa, Alabama 35401	4

11. Don Hinsman 1
c/o Department of Mathematics
Naval Postgraduate School
Monterey, California 93940